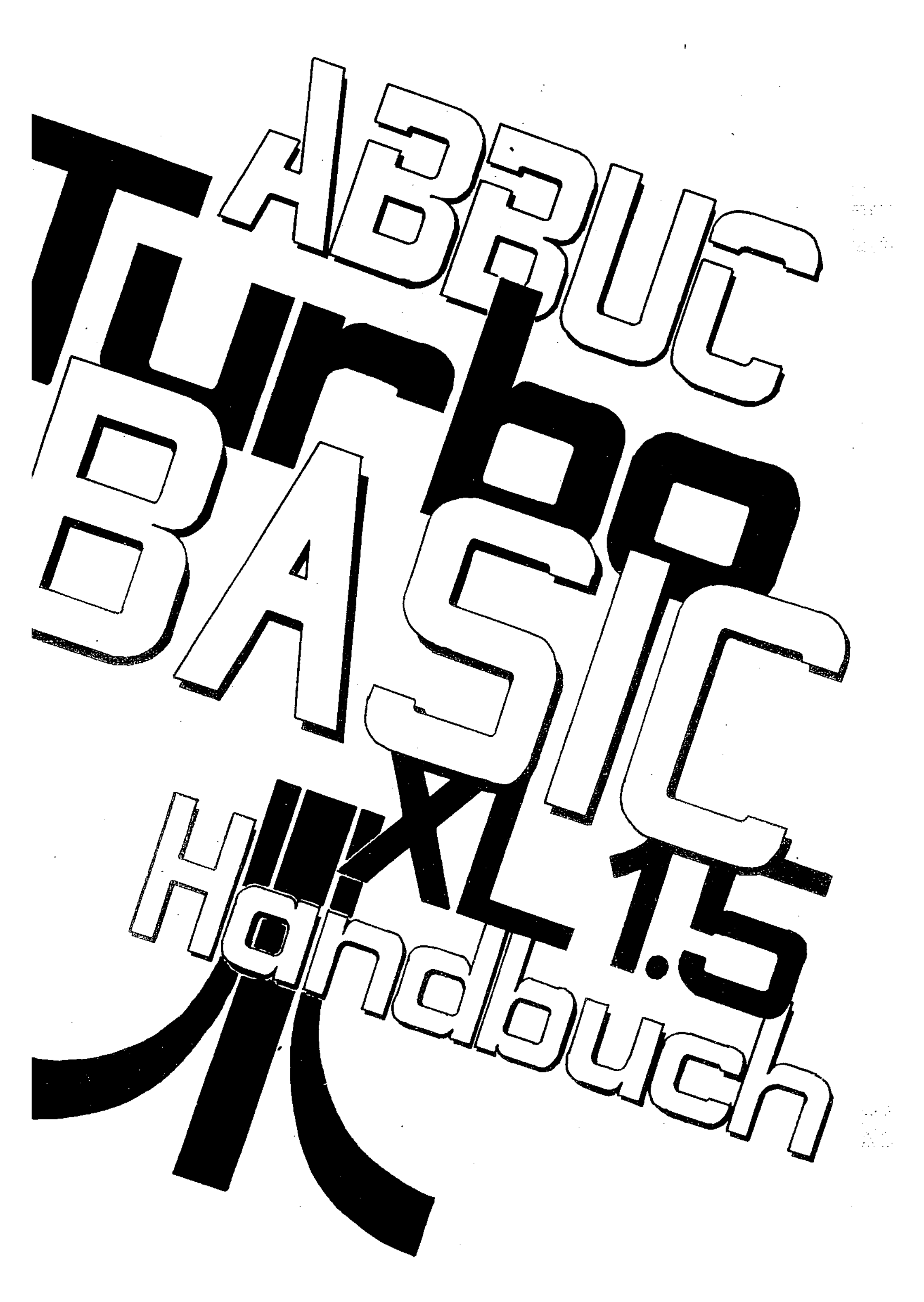

ABBUC
Turbo
BASIC
XL 1.5
Handbuch



Handbuch zum Turbo Basic
von Frank Ostrowski
Originaltitel:
TURBO BASIC XL 1.5
MANUAL
von Wil Braakman

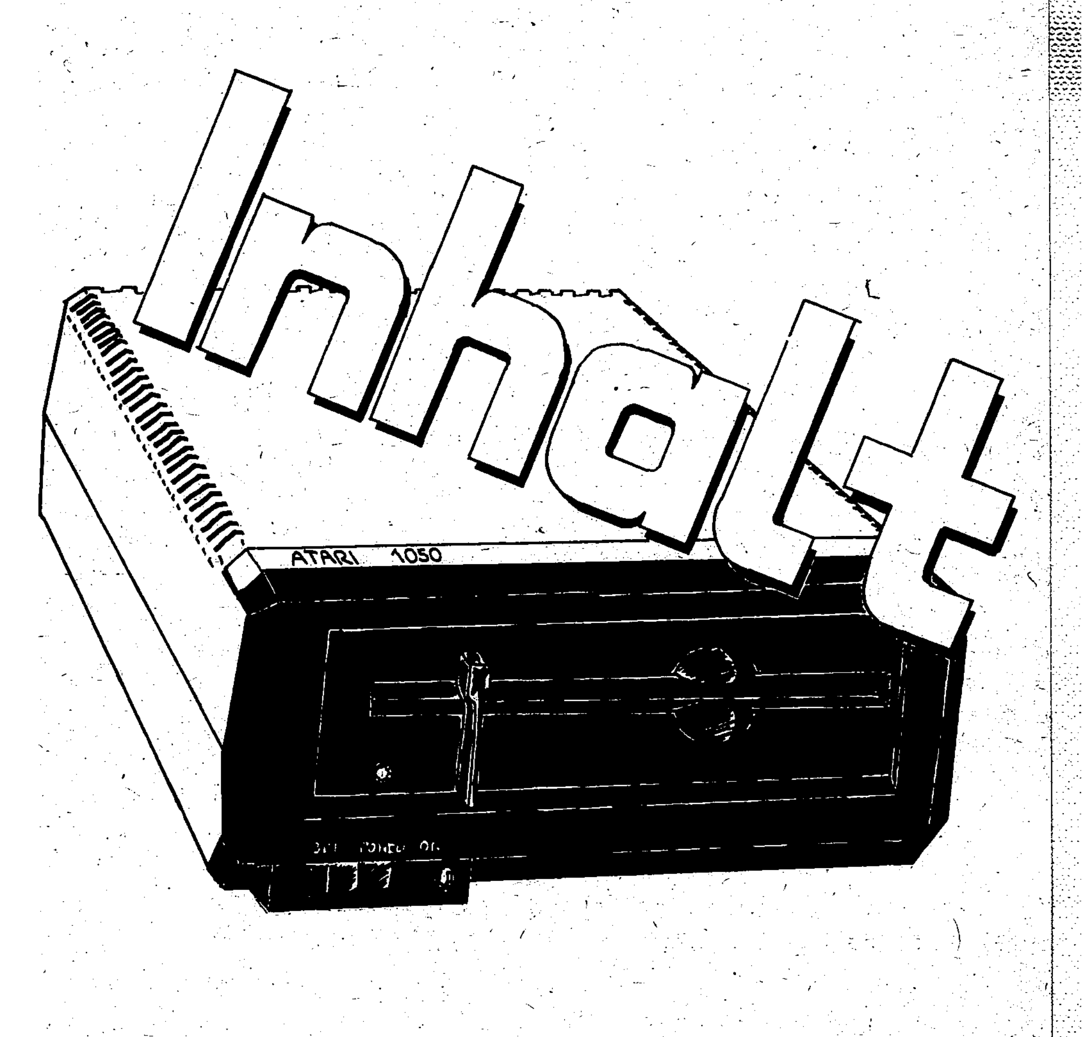
erschienen 1988 in den Niederlanden
bei SAG (Stichting Atari Gebruikers)

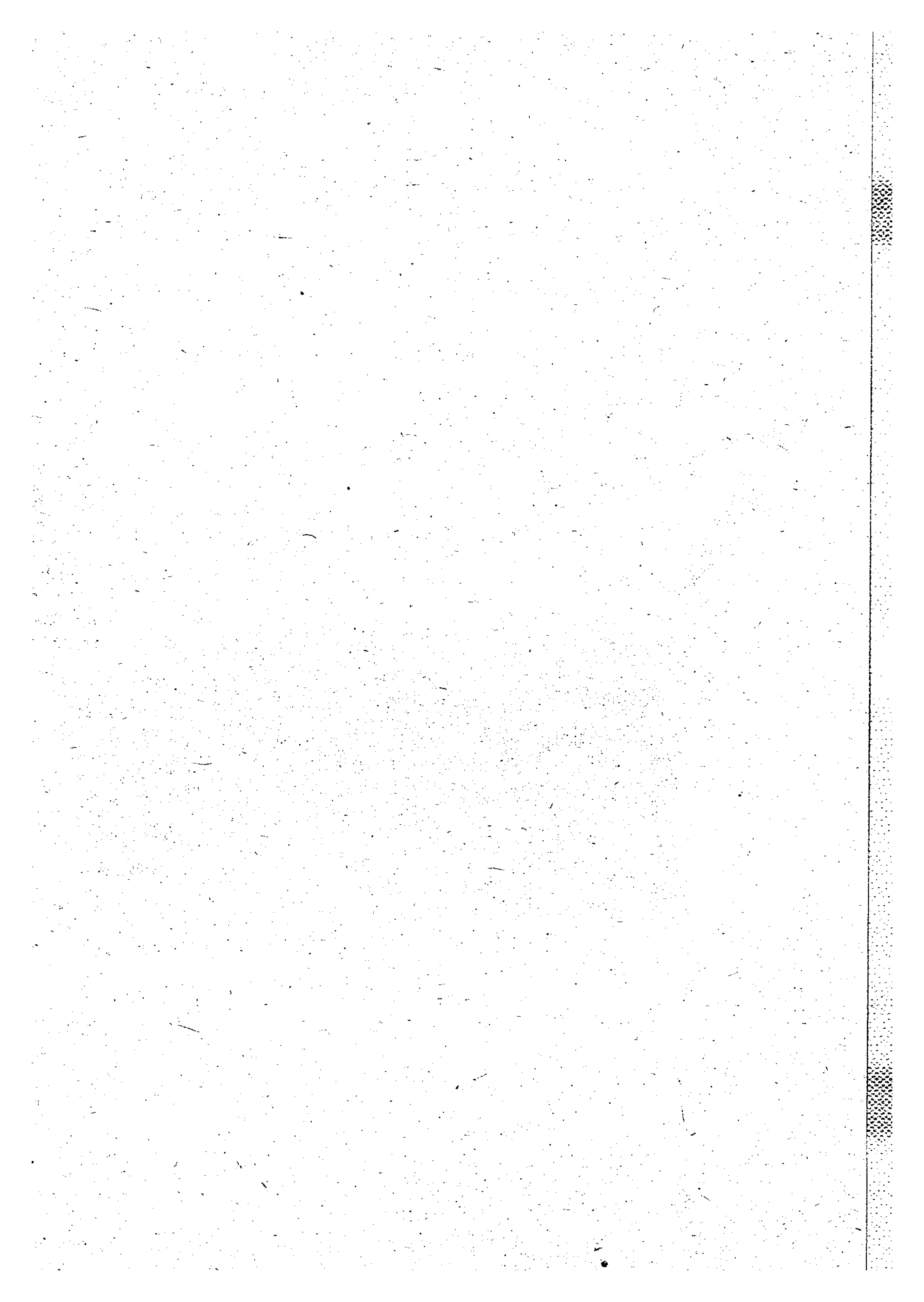
Deutsche Übersetzung: Rolf A. Specht 1990
Grafische Gestaltung: ANALOG RESEARCH
Rolf A. Specht 1990
Erschienen im Eigenverlag des
Atari Bit Byter User Club e.V.

Wir danken Eli Maas für die Unterstützung

A.B.B.U.C. e.V.
c/o Wolfgang Burger
Wieschenbeck 45
D-4352 Herten

Dieses Buch wurde allen Mitgliedern des A.B.B.U.C. e.V. als
Jahresgabe 1990 kostenlos überreicht.


Inhalt
ATARI 1050



TURBO BASIC XL 1.5 HANDBUCH

Inhalt

E i n l e i t u n g :

H a u p t t e i l 1 :

H a u p t t e i l 2 :

H a u p t t e i l 3 :

Anhang A:

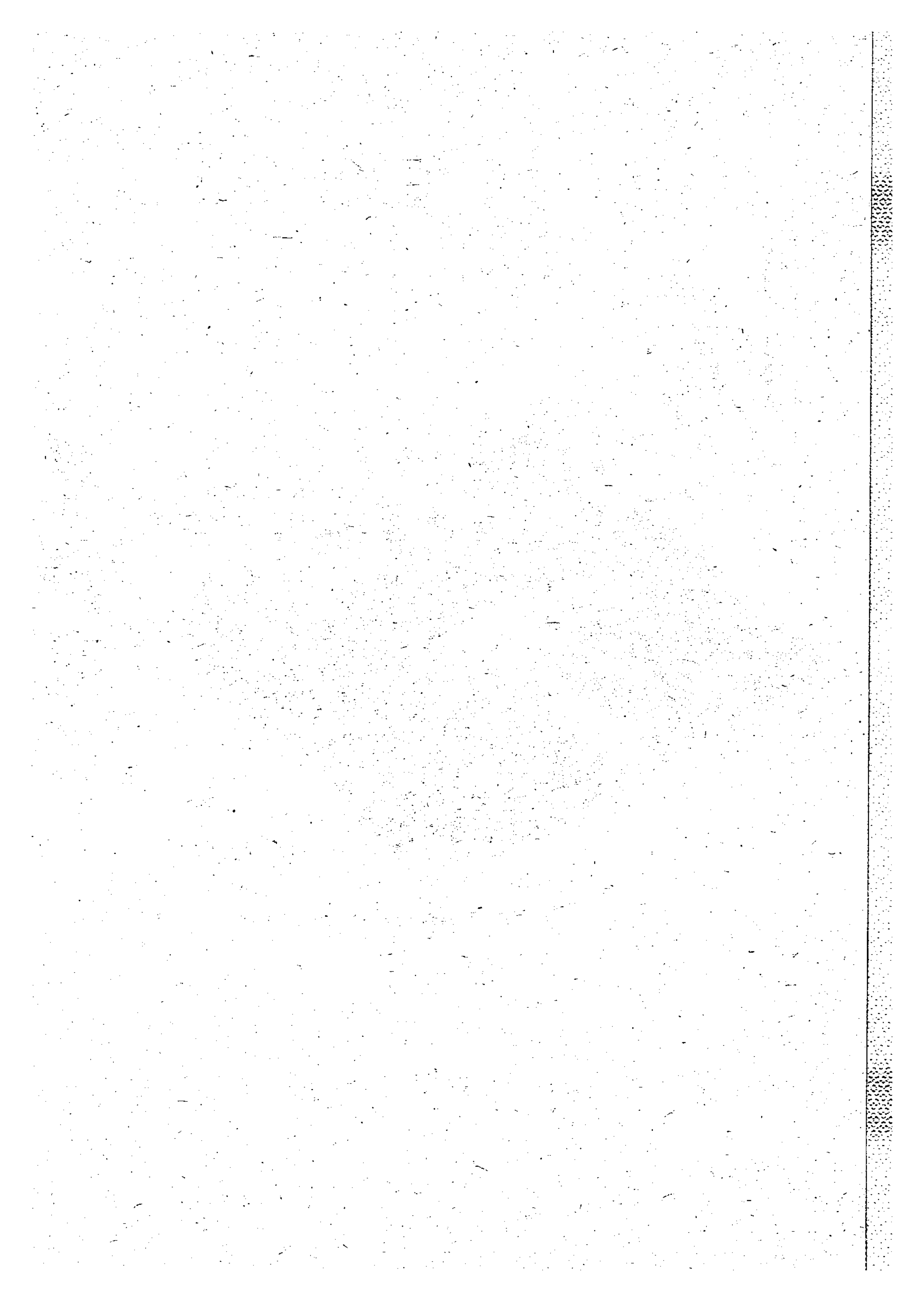
Anhang B:

Anhang C:

Anhang D:

Stichwortverzeichnis


Vorwort
Masterdisk
ATARI



Vorwort

Turbo BASIC XL 1.5 ist vom 24-jährigen Frank Ostrowski aus München 1985 entwickelt und von der deutschen Computerzeitschrift HAPPY COMPUTER im Dezember desselben Jahres veröffentlicht worden. Alle ATARI 8-Bit Anwender und Anwenderinnen und vor allem der Autor dieses Handbuches sind beiden sehr dankbar. Dem einen für die Entwicklung und der anderen für die Veröffentlichung.

Das Turbo BASIC XL 1.5 ist bis jetzt (1990) die schnellste BASIC Version für die ATARI 8-bit Computer mit mindestens 64 kBytes Speicher.
Durch die Anwendung dieser BASIC Version erhält man zudem zusätzliche Bytes freien Speicherplätzes. Im ATARI BASIC sind für den Programmierer und die Programmiererin 32274 Bytes frei, während es im Turbo BASIC XL 1.5 ganze 34017 Bytes sind.

Das Turbo BASIC XL 1.5 eignet sich sehr gut, um strukturiert zu programmieren. Der unübersichtliche Sprungbefehl GOTO kann nun getrost vergessen werden. Schleifenkonstruktionen, wie REPEAT....UNTIL, WHILE....WEND und DO....LOOP, sind viel geschickter, schneller und besser lesbar.

Angesichts der Tatsache, daß es für ATARI genügend Bücher mit Erläuterungen gibt, aber für Turbo BASIC XL 1.5 noch keine, habe ich beschlossen, hieran etwas zu ändern.

Das Ergebnis ist ein Handbuch (also kein LEHRBUCH), sowohl für ATARI BASIC (Rev. B und C) als auch für Turbo BASIC XL 1.5. Jeder Befehl, jede Anweisung und Instruktion wird in diesem Handbuch behandelt und die Syntax erklärt. Finden Sie eine Instruktion nicht in diesem Handbuch, dann existiert sie auch in keinem der beiden BASIC Dialekte.

Frank Ostrowski hat nicht nur dieses sehr intelligente und schnelle BASIC entwickelt. Er hat hierfür auch einen Compiler und ein Runtime Programm geschrieben.
Auch diese wurden von HAPPY COMPUTER veröffentlicht (HAPPY COMPUTER Sonderheft 1 02/86).
Das Turbo BASIC XL 1.5 besitzt ähnliche Befehle wie das MICROSOFT-BASIC und eine dem PASCAL ähnliche Struktur.
Menschen, die also mit einem IBM Personal Computer oder einem kompatiblen Rechner zu tun haben, werden in diesem Handbuch sicher Funktionen und Befehle wiederfinden, die sie beim Umgang mit einem der genannten Rechner gewohnt sind.
Ich hoffe, daß jeder Mensch, der sich durch dieses Handbuch gearbeitet hat, seinen Nutzen daraus ziehen kann und soviel Spaß mit der Anwendung desselben hat, wie ich damit bereits hatte.

Ich habe mit gewissen Vorkenntnissen vom Computern und Programmieren angefangen, das Handbuch zu schreiben, doch entdeckte ich immer wieder, daß man immer noch etwas dazu lernt.

Wil Braakmann, im April 1987

Wie man dieses Handbuch benutzt

Wenn man dieses Handbuch richtig benutzen will, sollte man doch einige Erfahrung mit den Grundprinzipien des Programmierens haben. Es ist nicht meine Absicht, Ihnen das Programmieren beizubringen.

Das Handbuch ist in 3 Hauptteile und einige Anhänge unterteilt.

Hauptteil 1: Gibt Ihnen eine Übersicht der BASIC Versionen für die ATARI Computer.

Hauptteil 2: Erklärt Ihnen kurz, wie Sie BASIC starten und anwenden.

Gibt allgemeine Informationen, die Sie wissen sollten, bevor sie mit dem Programmieren anfangen.

Hauptteil 3: Das Herz des Handbuches, das Nachschlagewerk. Es enthält die Syntax jeder Instruktion, Anweisung und Funktion, sowohl in ATARI BASIC als auch in Turbo BASIC XL 1.5 in alphabetischer Reihenfolge.

Die Anhänge enthalten andere nützliche Informationen, wie zum Beispiel eine vollständige Liste der Fehlermeldungen, ATASCII-Codes und so weiter.

Ich empfehle Ihnen, den Hauptteil 2 aufmerksam zu lesen, so daß Sie mit BASIC und den verwendeten Codes im Nachschlagewerk vertraut werden.
Danach können Sie den Schritt zum Hauptteil 3 wagen; dem "Nachschlagewerk". Hier finden Sie Informationen, wie jede Anweisung oder Funktion in Ihren Programmen verwendet werden muß.

Im Nachschlagewerk werden verschiedene Zeichen verwendet, die eine bestimmte Funktion/Bedeutung haben.

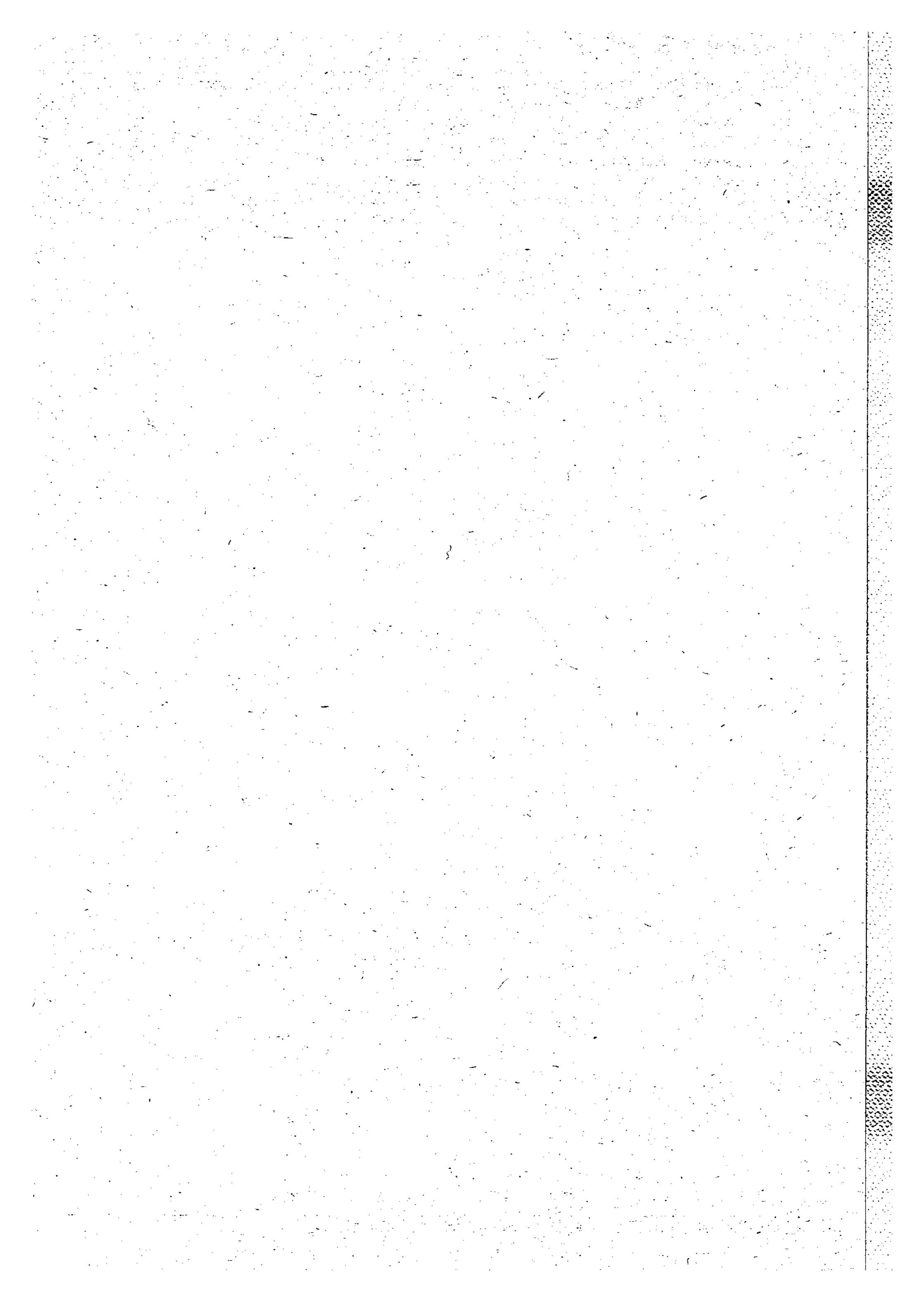
*** bei der Version bedeutet, daß die Instruktion oder Funktion in der betreffenden Sprache verwendet werden kann.

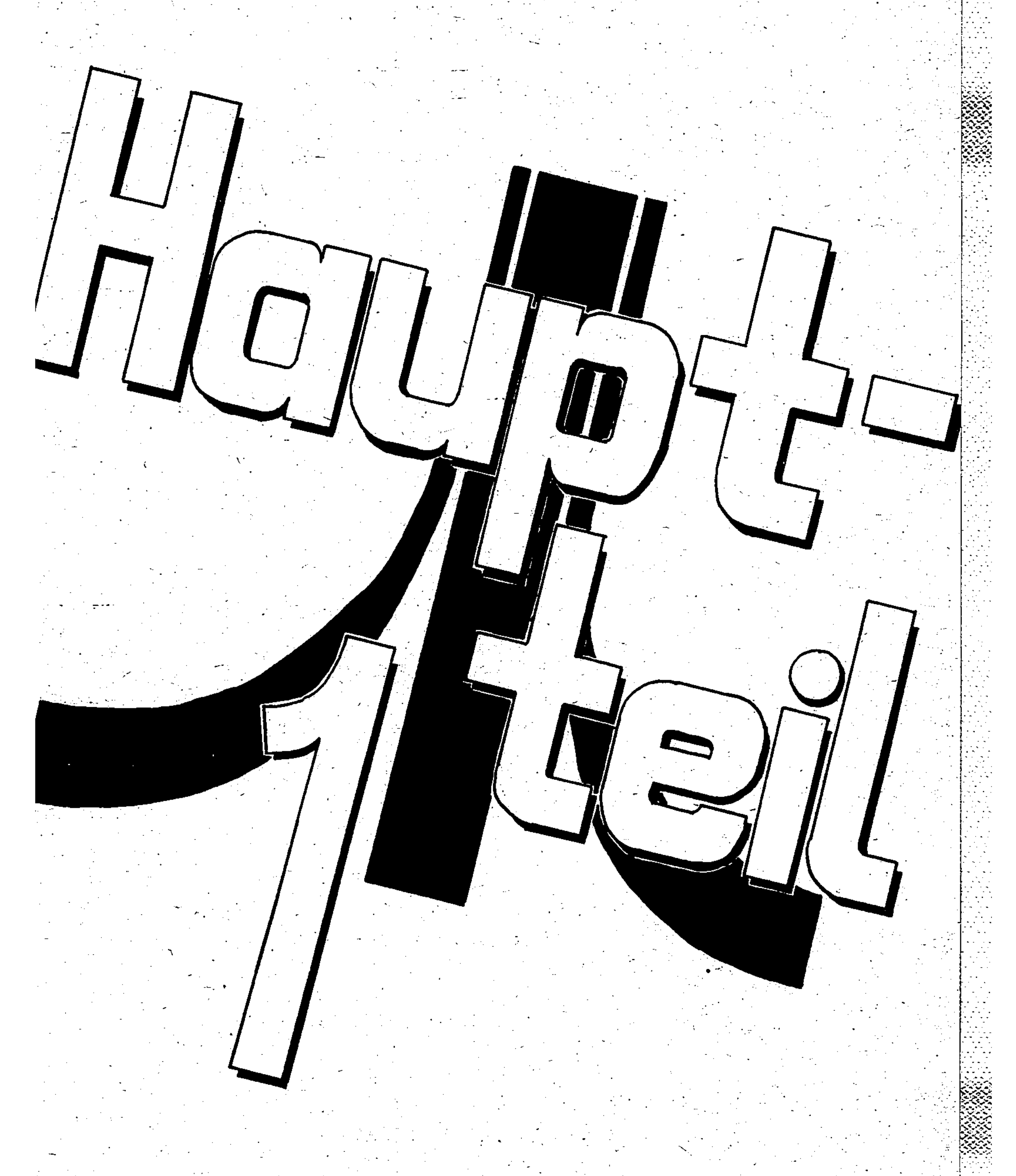
[ ] bedeutet, daß alles, was hier zwischen steht, beim Programmieren nicht unbedingt eingegeben werden muß.

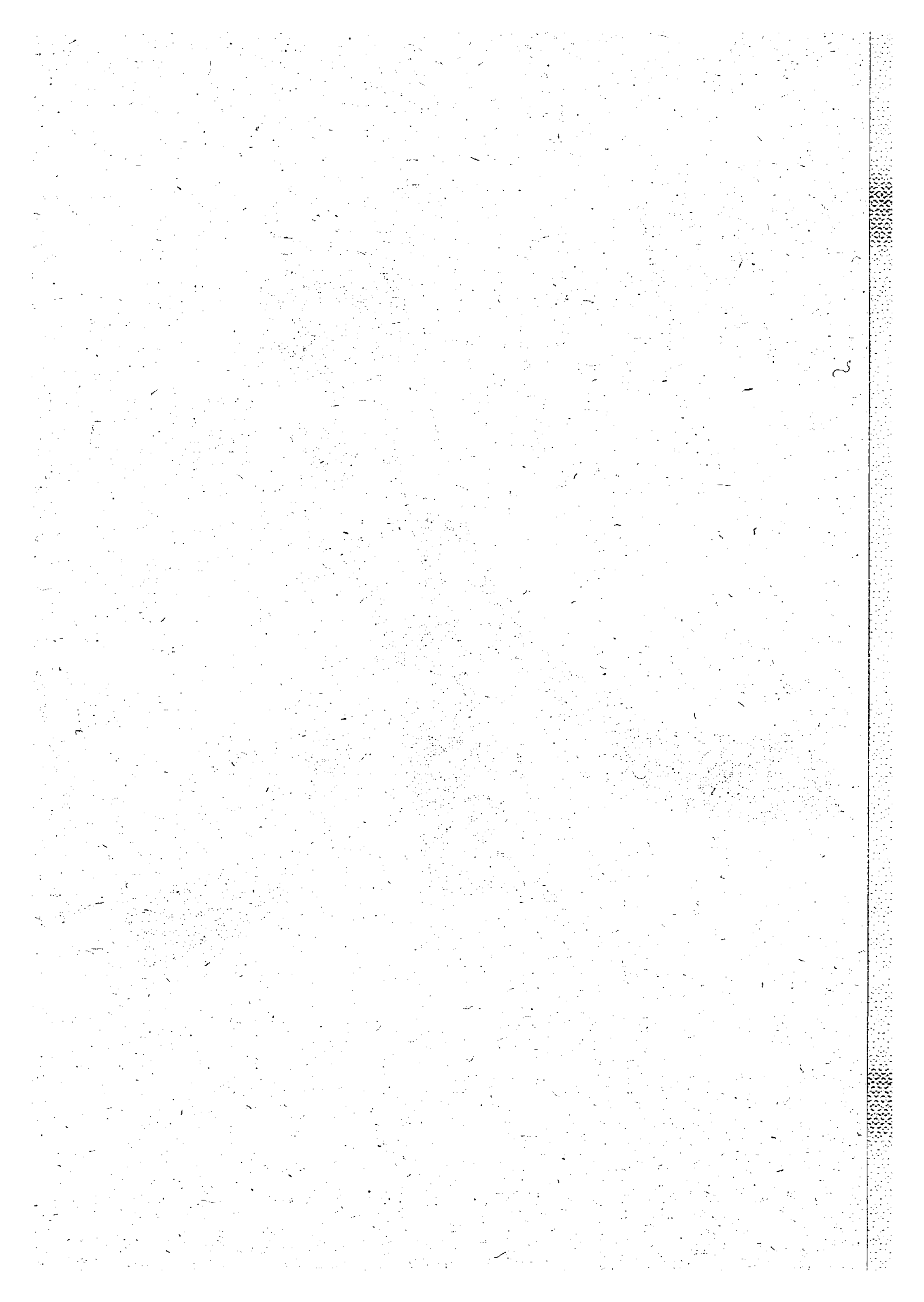
... bedeutet, daß eine Zuweisung, Variable oder anderes so oft, wie Sie wollen, wiederholt werden darf.

Beispiel: INPUT [#kanal,]var[,var] ...
Das heißt, daß INPUT die Anweisung ist. Ihr muß eine Variable folgen (das erste var). Falls gewünscht, darf auch ein Kanal, der mit OPEN spezifiziert worden ist, angesprochen werden.
Dann dürfen auch mehrere Variablen (das zweite var, usw.) be-

nutzt werden. Oder genauso mit der anderen INPUT Anweisung:
INPUT ["text"],var[,var] ...
Der Anweisung INPUT muß eine Variable folgen (das erste var). Der Text und die weiteren Variablen sind nicht zwingend vorgeschrieben. Kommata und Anführungszeichen zwischen [ ] sind vorgeschrieben, sofern sie dort verwendet werden. Falls Sie beim Arbeiten mit Hauptteil 3 ein benutztes Zeichen nicht verstehen, dann schauen Sie sich stets die untenstehenden Beispiele an.




Haupt-
teil
1



Hauptteil 1

Die BASIC Versionen.

Für die ATARI 8-Bit Computer sind momentan folgende BASIC Versionen (Dialekte) verfügbar:

ATARI BASIC Rev. A, B und C
Turbo BASIC XL 1.5
BASIC XE (Normal)
BASIC XE (Schnell)
Action
MMG-Compiler
ABC-Compiler

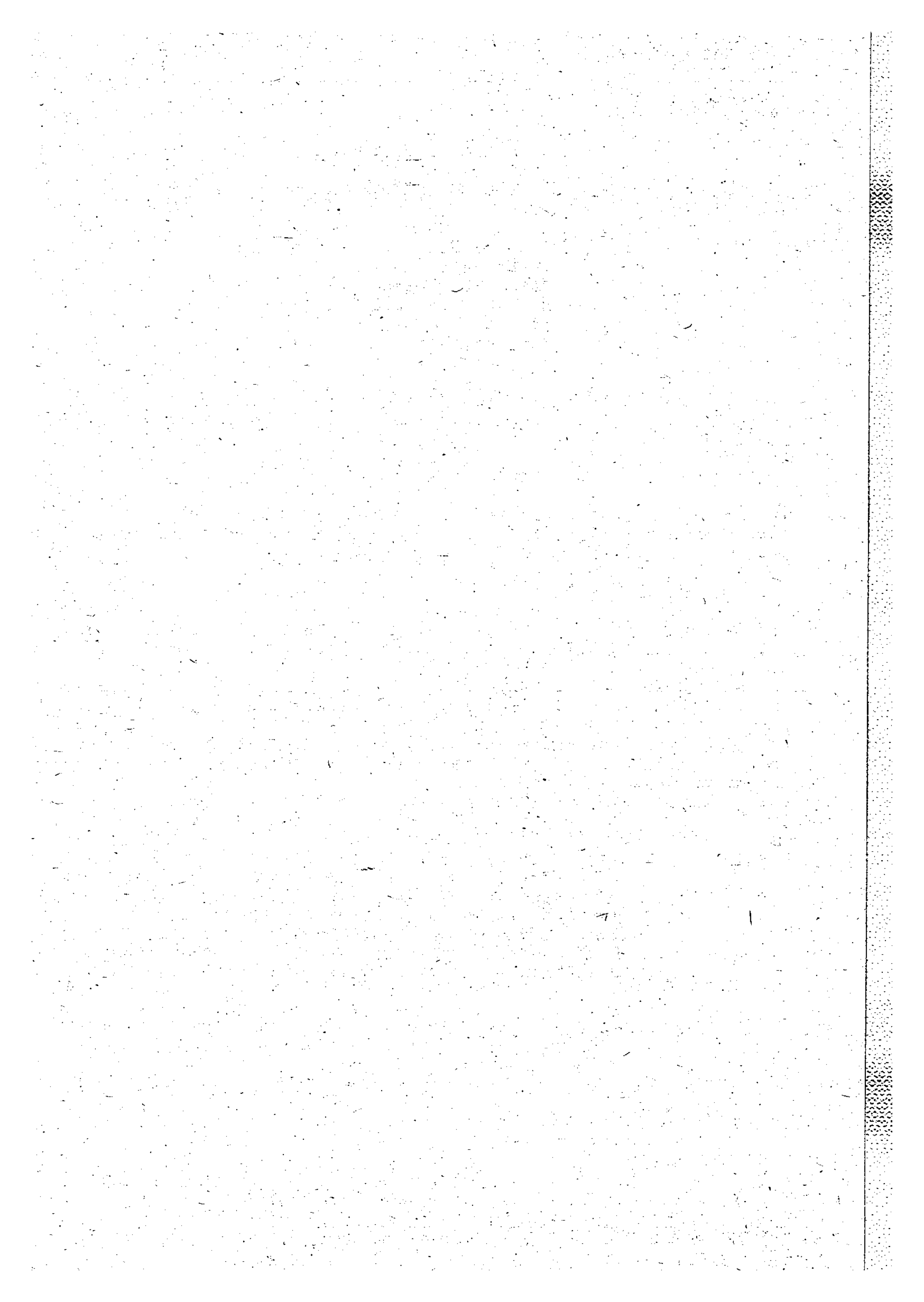
Die letzten beiden sind eigentlich keine Computersprachen, sondern Compiler (Programme, die BASIC Programme in Maschinensprache übersetzen).
In diesem Handbuch werden nur folgende BASIC Versionen behandelt: ATARI BASIC und Turbo BASIC XL 1.5.

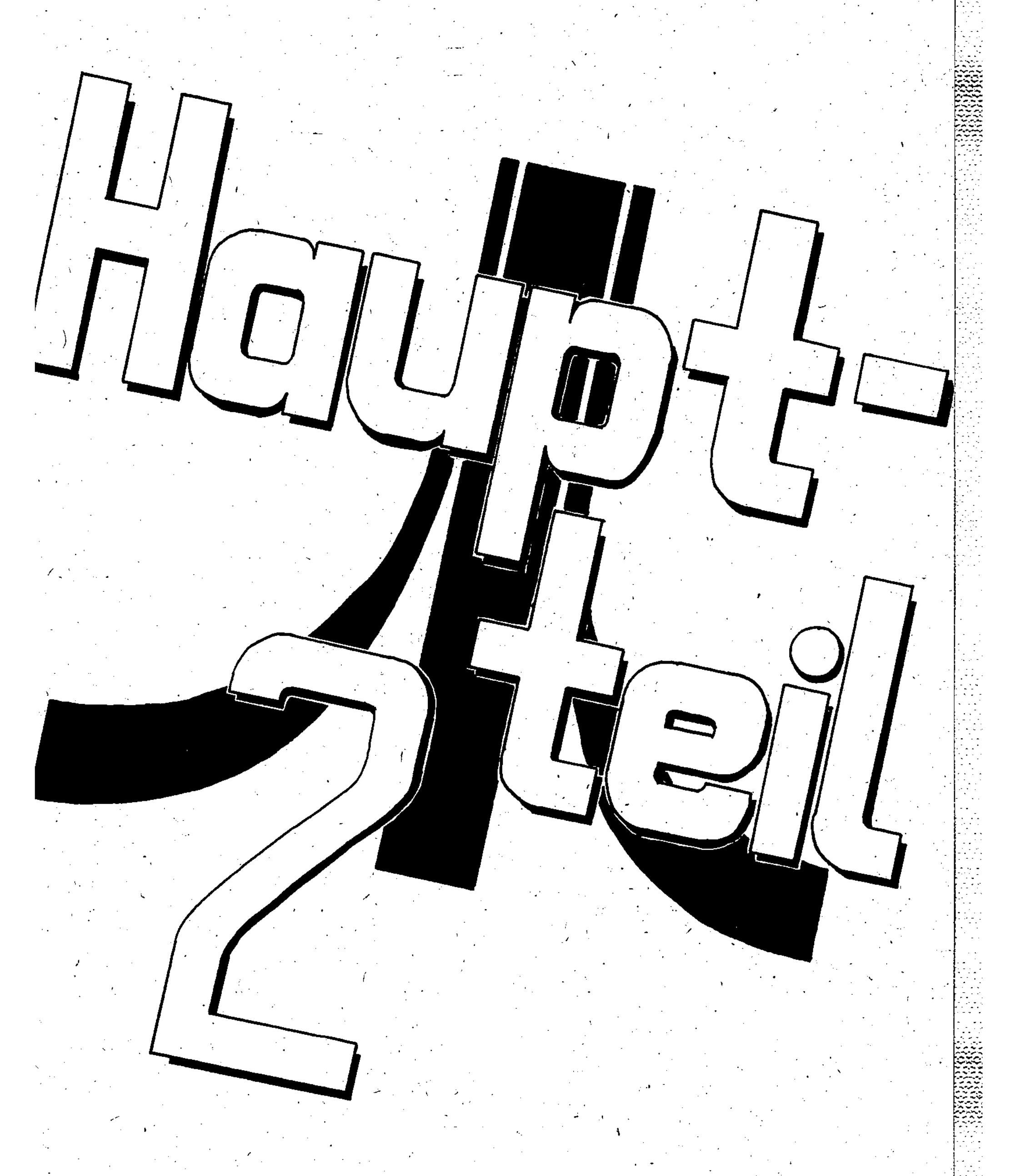
Im Hauptteil 3 wird erklärt, in welchem der beiden Dialekte eine Instruktion gültig ist, und ob sie auch beim Kompilieren benutzt werden darf. Dies wird dann wie folgt angezeigt:

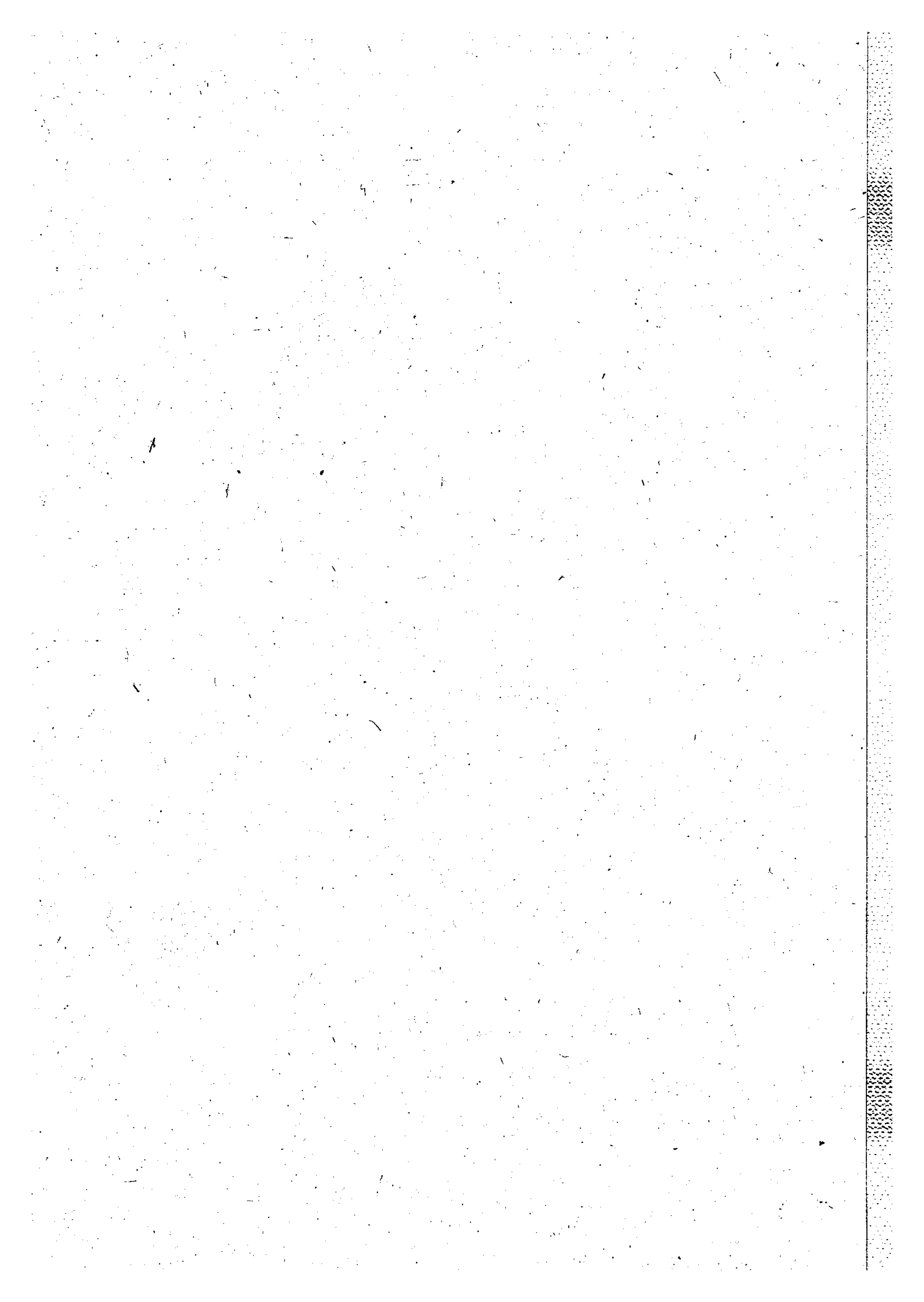
Version:	ATARI BASIC	Turbo BASIC	Compiler
	***	***	***

Die Sternchen (***) zeigen an, daß die betreffende Instruktion tatsächlich im jeweiligen BASIC Dialekt oder mit dem Compiler benutzt werden kann.
Es sei hier nochmals angemerkt, daß zum Betrieb von Turbo BASIC XL 1.5 (mindestens) ein Diskettenlaufwerk und ein Computer mit (mindestens) 64 kByte Speicher nötig sind.
Es gibt wohl auch eine Anpassung an den Kassettenrekorder, doch in dieser "abgespeckten" Version verliert Turbo BASIC viele seiner guten Eigenschaften.
Darum wurde diese Kassettenversion hier bewußt weggelassen und nicht weiter besprochen. Wer ernsthaft in BASIC programmieren will, kann um ein Diskettenlaufwerk nicht herum.
Spezielle Anwendungen in BASIC mit Diskettenlaufwerk sind:

- *Input/Output zur und von der Diskette im Gegensatz zum Kassettenrekorder. Ein Diskettenlaufwerk ist nämlich ein adressierbares Speichermedium.
  Sehen Sie sich dazu auch Anhang B, "Spezielle Anwendungen von Input/Output (Eingabe-/Ausgabe-) Operationen beim Gebrauch von Dateien in BASIC" an.
- *Schnellerer Transport der Daten vom und zum Speichermedium (Wodurch eine Datenbank im Gegensatz zu einer geschriebenen Notiz überhaupt erst brauchbar wird).




Haupt-
teil
2



Hauptteil 2

Wie starte ich BASIC?

<u>ATARI BASIC:</u>

Da Sie mit einem Diskettenlaufwerk arbeiten, gehen Sie wie folgt vor:

-Schalten Sie Ihr Datensichtgerät (Bildschirm) und Ihr Laufwerk ein. Die Reihenfolge ist hier nicht wichtig. Legen Sie nun eine Diskette in Ihr Laufwerk, auf der sich nur ein DOS befindet. Auf dieser Diskette ist genügend Platz für Programme, die Sie eventuell schreiben und speichern wollen. Benutzen Sie keine Diskette, auf der sich ein "Bootfile" (ein Maschinensprachenprogramm, das automatisch geladen und gestartet wird) befindet, denn sonst kommen sie nicht mehr ins BASIC.

-Schalten Sie danach Ihren Computer ein. Das Laufwerk mit der eingelegten Diskette wird dann anfangen zu arbeiten (Das "Busy"-Licht geht an). Auf diese Weise erkennt der Computer, daß ein Diskettenlaufwerk angeschlossen ist; dies ist wiederum wichtig, um von Diskette zu lesen und auf Diskette zu schreiben (Read/Write).

-Nach einigem Gesurre wird auf Ihrem Bildschirm das Wort READY erscheinen. Sie wissen dann, daß Sie sich im ATARI BASIC befinden. Die Befehle aus dem Hauptteil 3, wo drei Sterne (***) unter der Version ATARI BASIC stehen, können nun zur Anwendung kommen.

<u>Turbo BASIC XL 1.5:</u>

Um Turbo BASIC XL 1.5 starten zu können, müssen Sie erst einiges erledigen.

1) Starten Sie Ihre Masterdiskette DOS 2.0 oder DOS 2.5, wie oben beschrieben. Wenn READY auf dem Schirm erscheint, verlassen Sie BASIC und gehen ins DOS-Menü, indem Sie DOS eintippen und danach die RETURN-Taste drücken.

2) Im DOS-Menü wählen Sie die Option I und formatieren mit ihrer Hilfe drei leere Disketten.

3) Schreiben Sie mit der Option H die DOS-Dateien DOS.SYS und DUP.SYS auf jede der Disketten.

4) Legen Sie nun Ihre Masterdiskette Turbo BASIC XL 1.5 ins Laufwerk.

5)Waehlen Sie danach die Option O (Duplicate file/Datei duplizieren). Beantworten Sie die Frage "Which file to move?" (Welche Datei soll kopiert werden?) mit AUTORUN.SYS und drücken Sie die RETURN-Taste. Folgen Sie dann den Anweisungen des DOS, das Sie auf notwendige Diskettenwechsel aufmerksam macht.

6) Wiederholen Sie die Punkte 4) und 5) nochmals, aber geben Sie als DateinamenCOMPILER.COM an. Benutzen Sie als Zieldiskette eine der beiden übrig gebliebenen.

7) Wiederholen Sie die Punkte 4) und 5) ein letztes mal, geben Sie jedoch als DateinamenRUNTIME.OBJ an und benutzen Sie die übrig gebliebene Diskette als Ziel.

Sie haben nun vier Disketten:
- Ihre Masterdiskette (die Sie nun archivieren sollten).
- Eine Diskette mit der DateiAUTORUN.SYS.Dies ist Ihre Turbo BASIC XL 1.5 Arbeitsdiskette.
- Eine Diskette mit der Datei COMPILER.COM. Mit dieser Diskette können Sienun BASIC-Programme kompilieren.
- Eine Diskette mit der Datei RUNTIME.OBJ. Auf dieser Diskette sollten sich Ihre kompilierten Programme befinden, um gestartet werden zu können.

Beschriften Sie nun Ihre Disketten, damit es zu keinen Verwechslungen kommt!

8) Waehlen Sie nun die Option E des DOS-Menüs.

9) Legen Sie nun die Diskette mit COMPILER.COM ein und drücken Sie die RETURN-Taste. Tippen Sie danach COMPILER.COM,AUTORUN.SYS ein und drücken Sie wieder die RETURN-Taste. Ihr Kompilierprogramm wird jetzt in AUTORUN.SYS umbenannt. Vergessen Sie bitte das Komma zwischen den Dateinamen nicht.

10) Wiederholen Sie Punkt 9) mit der Diskette, auf der RUNTIME.OBJ steht. Geben Sie nach der Wahl von Option E RUNTIME.OBJ,AUTORUN.SYS an.
Wenn alles gut gegangen ist, haben Sie jetzt drei formatierte Disketten mit jeweils einer Datei AUTORUN.SYS. Kontrollieren Sie das mit der Option A des DOS-Menüs. Das heißt: Tippen Sie A und drücken Sie zweimal die RETURN-Taste.

Nun sind alle Vorkehrungen getroffen, um Turbo BASIC XL 1.5 zu starten. Es gibt dafür zwei Möglichkeiten, als da wären:

a) Über das DOS-Menü mit der Option L. Geben Sie als Dateinamen AUTORUN.SYS an, und Turbo BASIC wird geladen und gestartet. Natürlich muß zu diesem Zweck die Arbeitsdiskette in Ihrem Laufwerk stecken.
b) Die elegantere Methode: Schalten Sie Ihren Computer aus und legen Sie die Turbo BASIC Arbeitsdiskette ins Laufwerk. Danach schalten Sie den Computer wieder ein. Das Turbo BASIC XL 1.5 wird dann geladen und gestartet, ohne daß die OPTION-Taste gedrückt werden muß.

Wenn READY auf dem Bildschirm erscheint, sind Sie im Turbo BASIC. Das können Sie überprüfen, indem Sie DIR tippen und die RETURN-Taste drücken. Das Inhaltsverzeichnis der Diskette wird dann auf dem Bildschirm erscheinen. Sie werden sehen, daß sich folgende Dateien auf der Diskette befinden:

DOS.SYS, DUP.SYS und AUTORUN.SYS.

DUP.SYS (das DOS-Menü) ist nun nicht mehr notwendig und kann gelöscht werden. Sie bewerkstelligen dies folgendermaßen: tippen Sie DELETE"D:DUP.SYS" und drücken Sie die RETURN-Taste. Prüfen Sie auch dies wieder mit DIR nach.
Sie haben nun schon Bekanntschaft mit Turbo BASIC XL 1.5 gemacht, und zwar durch den Gebrauch von DIR und DELETE, die DOS-Befehle sind. Es ist viel praktischer und direkter, Sie vom

BASIC aus ausführen zu können, als dazu erst ins DOS-Menü gehen zu müssen.
Sie können nun ein Programm eingeben oder ein Programm von Diskette laden und starten.
Das Laden und Starten eines BASIC-Programmes kann auf zwei Arten geschehen, und zwar:
a) Laden Sie erst mit LOAD"D:DATEINAM.ANH" ein Programm in den Speicher und starten Sie es dann mit RUN. Oder
b) starten Sie ein Programm von der Diskette, automatisch, nachdem es in den Speicher geladen wurde. Das geht mit RUN"D:DATEINAM.ANH".

AUSFÜHRUNGSARTEN

Wenn BASIC (ATARI BASIC oder Turbo BASIC) gestartet wurde, wird die Meldung READY auf dem Bildschirm gezeigt. READY (engl.: fertig, bereit) bedeutet, daß BASIC bereit ist, Ihre Instruktionen entgegenzunehmen und auszuführen. Dieser Zustand wird auch "command level" (engl.:Befehlsebene) genannt. Jetzt können Sie auf zwei verschiedene Arten mit BASIC kommunizieren: die direkte Weise oder die indirekte Weise.

INDIREKTE WEISE:

Wenn Sie ein Programm schreiben, dann tun Sie dies im indirekten Modus. Um BASIC mitzuteilen, daß Sie im indirekten Modus arbeiten, benutzen Sie Zeilennummern. Die Zeilennummern sind ein Bestandteil des Programms und werden deshalb mit dem Rest des Programms im Speicher aufbewahrt. Das im Speicher befindliche Programm kann mit dem RUN Befehl ausgeführt werden. Zum Beispiel so:

```
  READY
   10 PRINT 20+5
   RUN
   25
 READY
```

DIREKTE WEISE:

Direkte Weise bedeutet, daß Sie BASIC anweisen, den eingegebenen Befehl sofort auszuführen, sobald die RETURN-Taste gedrückt wird. Das machen Sie BASIC deutlich, indem Sie die Zeilennummern vor den Anweisungen, Befehlen und Instruktionen weglassen. Auf diese Art und Weise kann man den Computer sehr gut für kleine und größere Berechnungen benutzen. Die Ergebnisse werden vom Computer behalten, indem man Sie in Variablen setzt. Die Anweisungen usw. werden vom Computer nicht behalten. Der direkte Modus ist sehr nützlich, um Fehler in Ihren eigenen Programmen aufzuspüren. Ein Beispiel für eine Berechnung ist:

```
READY
 PRINT 20+5
 25
READY
```

Der Gebrauch von Variablen im direkten Modus geht wie folgt:

```
READY
 A=20: B=5: C=A+B: PRINT C
 25
READY
```

Falls Sie danach wissen wollen, welchen Wert jeweils A, B und C haben, brauchen Sie nur das folgende einzugeben:

```
READY
 PRINT A,B,C
 20      5      25
READY
```

Das Wort READY soll in den obenstehenden Beispielen nicht mit eingetippt werden. READY ist die Bereitschaftsmeldung des BASIC und wird automatisch auf den Schirm geschrieben.
Ein weiterer Vorteil von Turbo BASIC ist, daß für Anweisungen, Befehle und Instruktionen sowohl kleine als auch große Buchstaben benutzt werden dürfen. Sogar inverse Zeichen sind erlaubt. Turbo BASIC erkennt diese Zeichen und wandelt diese automatisch in Großbuchstaben um. Dies kann das normale ATARI BASIC nicht.

DER BASIC PROGRAMM EDITOR

Editor bedeutet Buchstaben-Verarbeiter oder einfach Schreiber. Der Einfachheit halber belassen wir es aber hier beim englischen Wort Editor. Jede beliebige Zeile, die man BASIC mitteilt, wird vom BASIC Programmeditor ausgeführt. Der Editor ist ein bildschirmorientierter Editor. Das heißt, Sie können auf dem Bildschirm etwas verändern, wo immer Sie wollen. Aber es kann nur jeweils eine Zeile gleichzeitig abgeändert werden. Die Änderung wird auch nur dann angenommen, wenn man in der jeweiligen Zeile die RETURN-Taste drückt.

Der Gebrauch dieser Editiermethode erspart einen Haufen Arbeit, wenn man bedenkt, daß es Computer gibt, die diese Möglichkeit nicht haben. Falls sich in der Zeile, die Sie gerade bearbeiten, ein Fehler befindet und Sie die RETURN-Taste schon gedrückt haben, müßten Sie die ganze Zeile nochmals korrigiert eingeben. Das wäre sehr viel Arbeit. Um mit dem Editor einigermaßen vertraut zu werden empfiehlt es sich, willkürlich einige Zeile einzugeben und einmal kreuz und quer mit dem Cursor (der kleinen, viereckigen Schreibmarke) über den Bildschirm zu flitzen.
Dies kann man tun, indem man die CONTROL-Taste gedrückt hält, während man eine der Pfeiltasten drückt (die Pfeiltasten befinden sich links neben der RETURN- und der CAPS-Taste.

SPEZIELLE FUNKTIONSTASTEN FÜR DEN EDITOR

Hiermit sind wir bei den für den Editor wichtigen Funktionstasten. Diese sind:

CONTROL zusammen mit einer der vier Pfeiltasten. Hiermit wird der Cursor bewegt, ohne daß irgendetwas geändert wird.

DELETE BACK SPACE Taste.

Hiermit bewegt sich der Cursor zurück (nach links) und löscht die Zeichen, die ihm entgegenkommen.

CONTROL zusammen mit INSERT. Hiermit wird der Text rechts vom und unter dem Cursor nach rechts bewegt, wobei eine Lücke mit Leerzeichen entsteht.

SHIFT zusammen mit INSERT fügt eine ganze Leerzeile ein. Alles was unter dem Cursor steht, wird nach unten gedrückt.

CONTROL zusammen mit DELETE BACK SPACE schiebt den Text zeichenweise von rechts nach links und löscht das Zeichen unter dem Cursor.

SHIFT zusammen mit DELETE BACK SPACE löscht die Zeile, in der sich der Cursor befindet und zieht die unteren Zeilen nach oben.

CAPS Taste.
Nach drücken von CAPS werden statt großer Buchstaben kleine ausgegeben. Durch nochmaliges drücken von CAPS wird dies wieder rückgängig gemacht.

CONTROL zusammen mit CAPS ruft die Grafikzeichen des ATARI Computers auf.

CONTROL zusammen mit 1 hält ein Programmlisting auf, daß über den Bildschirm läuft. Wenn nochmals CONTROL mit 1 gedrückt wird, läuft das Listing weiter.

RETURN Taste.
Dies ist die wahrscheinlich wichtigste Taste Ihres Computers. Sie schließen mit Ihr eine jeweilige Eingabe ab, ob das nun eine Programmzeile, ein Befehl oder eine Antwort auf eine Frage in einem Programm ist. Ein Beispiel: Sie werden nach einer Zahl gefragt und wollen 5 eingeben. Sie müssen danach die RETURN-Taste drücken, denn woher soll der Computer sonst wissen, daß Sie nicht 52 oder 539 eingeben wollen?

ATARI Taste (auf 800/400) / INVERSE VIDEO Taste (auf XL/XE Geräten)
Diese Taste liegt auf der Tastatur rechts unten. Hiermit sagen Sie dem Computer, daß ein Zeichen invers (Vorder- und Hintergrundfarbe vertauscht) dargestellt werden soll.

CONTROL oder SHIFT zusammen mit CLEAR hat zur Folge, daß der Bildschirm gelöscht wird. Die Programmzeilen, die schon mit RETURN abgeschlossen wurden, bleiben im Speicher.

Ferner gibt es im Turbo BASIC XL 1.5 noch einige Befehle, um bestimmte Editorfunktionen zu aktivieren. So gibt es zum Beispiel die Befehle DELETE und RENUM, LIST und NEW. Sehen Sie hierzu bitte im Hauptteil 3 im alphabetischen Nachschlagewerk nach.

Stets reagiert der Interpreter (der ständige BASIC-übersetzer) praktisch sofort, wenn Sie irgendwelche Fehler machen. Fehler, die die Syntax (den Aufbau) eines Befehles oder einer Anweisung betreffen, werden prompt mit einer ERROR (Fehler) Meldung quittiert. Der Cursor steht dann an der Stelle, die für den Interpreter unverständlich ist.
Fehler, die die Ausführung eines Programms betreffen, werden erst angezeigt, nachdem RUN eingegeben wurde. Und auch dies nur

dann, wenn das Programm den beteffenden Fehler ausführen muß.

Ein oft gemachter Fehler ist, daß vergessen wird, eine Stringvariable mit DIM zu dimensionieren. Das System gibt dann einen ERROR 9 aus. Wenn Sie die vollständige Liste der Fehlermeldungen studieren wollen, dann sehen Sie sich bitte den Anhang A an.

Zum Schluß sei noch angemerkt, daß man eine Programmzeile ebenfalls editieren kann, indem man sie mit LIST gefolgt von der Zeilennummer aufruft und danach verändert. In einer fehlerhaften Programmzeile steht am Anfang stets ERROR - usw., usw. .Dieser Fehler sollte beseitigt werden, andernfalls wird der Interpreter wieder eine Fehlermeldung ausgeben. Den Fehler behebt man, indem man den Cursor hinter die Zeilennummer bewegt und so lange CONTROL zusammen mit DELETE BACK SPACE drückt, bis die richtige Information in der Zeile übrigbleibt. Bestätigen Sie die Zeile dann mit der RETURN-Taste.

ALLGEMEINE INFORMATIONEN ÜBER DAS PROGRAMMIEREN IN BASIC

ZEILENFORM:

Programmzeilen haben in BASIC die folgende Form:

nnnnn BASIC Anweisung [:BASIC Anweisung...][:. Kommentar...]

BASIC-Programmzeilen werden mit der RETURN-Taste abgeschlossen. Dieses Format wird noch ausführlich besprochen.

Zeilennummern: "nnnnn" gibt die Zeilennummer an, die bis zu fünf Stellen lang sein darf. Jede BASIC Programmzeile beginnt mit einer Zeilennummer. Diese Zeilennummern werden gebraucht, um die Reihenfolge, wie die Zeilen im Speicher aufbewahrt werden, wiederzugeben. Außerdem dienen sie als Orientierung zum Editieren. Die Zeilennummern dürfen zwischen 0 und 65535 (ATARI BASIC:32767) liegen.

BASIC Anweisung: Eine BASIC Anweisung ist entweder ausführbar oder nicht ausführbar. Ausführbare Anweisungen sind Programminstruktionen, die BASIC mitteilen, was sie tun müssen, wenn das Programm ausgeführt wird. Zum Beispiel:PRINT A ist eine ausführbare Anweisung. Nicht ausführbare Anweisungen sind unter anderem DATA und REM, die keine Tätigkeit von BASIC erwarten, wenn BASIC auf sie trifft. Alle BASIC Anweisungen werden im folgenden Hauptteil ausführlich erklärt.

Falls gewünscht, dürfen mehrere Anweisungen (Instruktionen/Zuweisungen) in einer Zeile benutzt werden. Diese müssen dann mit einem Doppelpunkt [:] voneinander getrennt werden. Die maximale Länge einer Zeile darf 128 Zeichen nicht überschreiten. Sie werden vorher von einem Signalton gewarnt. Alles, was über 128 Zeichen hinausgeht, wird vom BASIC Interpreter nicht mehr beachtet. Ein Beispiel für "multiply statements" (Vielfache Anweisungen) :

```
READY
 10 FOR X=1 TO 5: PRINT X: NEXT X
 RUN
 1
 2
 3
 4
 5
READY
```

Kommentar: Kommentare (engl.: Remarks) dürfen am Ende einer Programmzeile angefügt werden, indem man einen Punkt [.] oder REM voranstellt, um BASIC deutlich zu machen, daß der Kommentar nicht beachtet werden soll und vom Vorhergehenden getrennt ist.

ZEICHENSATZ:

Der BASIC Zeichensatz (die Gesamtheit aller darstellbaren Zeichen) besteht aus alphabetischen Zeichen, numerischen Zeichen, Spezialzeichen und Grafikzeichen. Dies sind die Zeichen, die von BASIC erkannt werden.

Die alphabetischen Zeichen sind die Groß- und Kleinbuchstaben des Alphabets. Die numerischen Zeichen sind die Ziffern 0 bis 9.

Die folgenden Zeichen (Spezialzeichen) haben eine bestimmte Bedeutung für BASIC.

ZEICHEN	NAME/BEDEUTUNG
	Leerschritt (Leerzeichen)
=	Gleichheitszeichen
+	Pluszeichen
-	Minuszeichen
*	Multiplikationszeichen oder Stern
/	Divisionszeichen oder Schrägstrich
\	"Backslash"
^	Potenzzeichen
(	Klammer auf
)	Klammer zu
#	Nummerzeichen oder Kanalanzeiger
$	Dollarzeichen oder Stringanzeiger
!	Ausrufezeichen
&	Und
%	Prozentzeichen oder Integeranzeiger
,	Komma
.	Punkt oder Dezimalpunkt oder Kommentaranzeiger
;	Semikolon oder Platzbestimmung für die folgende Anweisung
'	Apostroph
:	Doppelpunkt oder Anweisungstrenner
?	Fragezeichen Oder PRINT Anweisung
@	Klammeraffe (Spezialzeichen)
"	Anführungszeichen oder Textbegrenzer
<	Kleiner als
>	Größer als
_	Unterstreichungszeichen

Dies ist allerdings nur eine Auswahl aus der großen Zahl der Zeichen, die die ATARI-Tastatur besitzt. Für eine vollständige Lis-

te aller Zeichen sehen Sie bitte in den Anhang C (ATASCII Zeichen).

RESERVIERTE AUSDRÜCKE

Sowohl in ATARI BASIC als auch in Turbo BASIC gibt es bestimmte Ausdrücke oder Wörter mit einer speziellen Bedeutung für das BASIC. Diese werden auch "reservierte Ausdrücke/Wörter" genannt. Die reservierten Ausdrücke umfassen alle BASIC Befehle, Anweisungen, Funktionen und Operatoren. Diese können nicht als Variablennamen benutzt werden, außer wenn die LET Anweisung benutzt wird.

Es wird jedoch davon abgeraten, reservierte Ausdrücke mit LET zu Variablennamen zu machen, da es die Lesbarkeit des Programms nicht fördert.

Unmseitig finden Sie eine Liste der reservierten Ausdrücke in BASIC. Falls diese nur im Turbo BASIC XL 1.5 benutzt werden, steht ein Stern dahinter.

ABS
ADR
ASC
ATN
*B [*]
BGET [*]
BLOAD [*]
BPUT [*]
BRUN [*]
BYE
CHR$
CIRCLE [*]
CLOAD
CLOG
CLOSE
CLR
CLS [*]
COLOR
COM
CONT
COS
CSAVE
DATA
DEC [*]
DEG
DEL
DELETE [*]
DIM
DIR [*]
DIV [*]
DO [*]
DOS
DPEEK [*]
DPOKE [*]
DRAWTO
DSOUND [*]
DUMP [*]
ELSE [*]
END
ENDPROC [*]
ENDIF [*]
ENTER
ERL [*]
ERR [*]
EXEC [*]
EXIT [*]
EXP
*F [*]
FCOLOR [*]
FILLTO [*]
FOR
FRAC [*]
FRE
GET
%GET [*]
GO# [*]
GOSUB
GOTO
HEX$ [*]
IF
INKEY$ [*]
INPUT
INSTR [*]
INT
*L [*]
LEN
LET
LIST
LOAD
LOCATE
LOCK [*]
LOG
LOOP [*]
LPRINT
-- [*]
MOD [*]
MOVE [*]
#NAME [*]
NEW
NEXT
NOTE
ON
OPEN
PADDLE
PAINT [*]
PAUSE [*]
PEEK
PLOT
POINT
POKE
POP
POSITION
PRINT
PROC [*]
PTRIG
PUT
%PUT [*]
RAD
RAND [*]
READ
REM
RENAME [*]
RENUM [*]
REPEAT [*]
RESTORE
RESTORE# [*]
RETURN
RND
RUN
SAVE
SETCOLOR
SGN
SIN
SOUND
SQR
STATUS
STICK
STOP
STR$
STRIG
TEXT [*]
TIME [*]
TIME$ [*]
TRACE [*]
TRAP
TRAP# [*]
TRUNC [*]
UINSTR [*]
UNLOCK [*]
UNTIL [*]
USR
VAL
WHILE [*]
WEND [*]

VARIABLEN

Variablen sind Namen, denen ein bestimmter Wert zugewiesen wird. Nach dem Inhalt der Variablen unterscheidet man zwischen numerischen und Stringvariablen. Eine numerische Variable enthält immer einen Zahlenwert, während eine Stringvariable ein Zeichen enthält, das eine Ziffer, ein Buchstabe oder ein grafisches Zeichen sein darf.

Die Länge der Variablen darf maximal 130 Zeichen lang sein, denn dies ist die maximale Größe des Eingabepuffers. Der Inhalt der Variablen darf so groß sein, wie freier Speicherplatz zur Verfügung steht.

Alle Variablen werden nach RUN oder CLR auf Null gesetzt. Falls eine numerische Variable nicht spezifiziert wurde, ist ihr Wert gleich Null, bei Stringvariablen gilt das gleiche.
Stringvariablen müssen vorab dimensioniert werden, das heißt, ihre Länge muß vorab angegeben werden (hiervon später mehr).

WIE MAN VARIABLEN BENENNT

BASIC Variablennamen dürfen so lang sein, wie Sie wünschen. Es ist jedoch besser, im Zusammenhang mit der maximalen Größe des Eingabepuffers nicht mehr als 130 Zeichen zu benutzen. Die Zeichen, die für einen Variablennamen benutzt werden, können von Ihnen frei bestimmt werden, die einzige Einschränkung ist, daß der Name mit einem Buchstaben anfangen muß. Ebenso dürfen Sie keinen Gebrauch von reservierten Ausdrücken/Wörtern machen. Sehen Sie sich dazu bitte die Liste der reservierten Ausdrücke/Wörter an, die in diesem Hauptteil aufgeführt wurde.
Auch der Anfang eines Variablennamen darf nicht einem reservierten Ausdruck/Wort beginnen. Beispiel:

```
10 ENDE=10
```

Dies ist nicht erlaubt, denn END ist ein reservierter Ausdruck in BASIC. Diese Zeile wird eine Fehlermeldung erzeugen.
Erlaubt ist hingegen:

```
10 DASENDE=10
```

Das gleiche gilt auch für Stringvariablen.

WIE MAN WERTE AN VARIABLEN ÜBERGIBT

Wichtig ist der Variablentyp: dies kann eine String- oder numerische Variable sein.

Stringvariablen wird ein Dollarzeichen [$] nach dem letzten Zeichen angefügt. Zum Beispiel:

```
A$="Turbo BASIC XL 1.5"
```

Das Dollarzeichen ist ein Deklarationszeichen. Es zeigt also, daß die Variable vom Typ "String" ist.
Will man Werte, so wie im obenstehenden Beispiel, an A$ übergeben, dann muß man erst dem internen Speicher mitteilen, daß ein

Fach geöffnet werden muß, und wie lang (wie groß) es sein soll. Dies funktioniert mit der DIM-Anweisung (DIM kommt von Dimensionierung; das heißt: einer Variable eine bestimmte Dimension geben). Um keine Fehlermeldung zu erhalten, teilen wir dem Computer also erst mit, wie groß das Fach A$ sein soll. In unserem Beispiel also 18 Zeichen.
Es reicht deshalb aus, A$ auf eine Länge von 18 zu dimensionieren. Nun ein Beispiel, wie das aussehen kann:

```
DIM A$(18): A$="Turbo BASIC XL 1.5"
```

Falls eine Stringvariable vorher nicht dimensioniert wurde, folgt die Fehlermeldung ERROR- 9.

Numerische Variablen brauchen im Gegensatz zu Stringvariablen nicht dimensioniert zu werden. Im Turbo BASIC wird nur ein Unterschied zwischen numerischen Variablentypen vom Typ Integer oder vom Typ "single precision" (Einfache Genauigkeit) gemacht.

Numerische Variablen einfacher Genauigkeit sind alle denkbaren Namen/Ausdrücke, die mit einem Buchstaben beginnen und nicht mit einem Dollarzeichen [$] abgeschlossen werden. Zum Beispiel:

```
A=10
JAN=15
VARIABLE2=11
PQR347AB=100
```

A, JAN und so weiter sind alles numerische Variablen. Es sei auch angemerkt, daß keine reservierten BASIC-Ausdrücke/Wörter benutzt werden dürfen (die Liste kennen Sie ja jetzt) .

Numerische Variablen vom Typ Integer können nur in Turbo BASIC zuerkannt werden. Hierzu fügen Sie das Prozentzeichen [%] vor den Variablennamen:

```
A=1.5734
PRINT %A
1
```

Im Gegensatz zu Stringvariablen können numerische Variablen in eine Tabelle/Folge aufgenommen werden. Diese Tabelle kann ein- oder zweidimensional sein. Um dies realisieren zu können, muß dem Computer erst wieder mitgeteilt werden, wie groß die Tabelle (auch Array genannt; engl.: Feld) sein soll.

Eindimensionale Felder (Arrays).

```
10 DIM A1(5)
```

Dies richtet ein eindimensionales Feld mit dem Namen A1 ein. Alle Elemente sind numerische Werte einfacher Genauigkeit. Das Feld hat eine Länge von 6 Fächern, denn Fach 0 zählt ja auch mit (man kann sich das auch mit Schubladen einer Kommode vorstellen). Alle Elemente haben in Turbo BASIC den Wert 0, wenn Sie dimensioniert werden.

Zweidimensionale Felder (Arrays).

```
10 DIM A2(2,3)
```

Dies richtet ein zweidimensionales Feld mit dem Namen A2 ein. Alle Elemente sind wieder vom Typ "Einfache Genauigkeit" und haben den Wert 0. (2,3) gibt an, wie groß die Tabelle/das Feld ist. (2,3) bedeutet, daß es 3 Reihen und 4 Spalten sein sollen (denken Sie daran, daß die 0 mitzählt!).

In einer Grafik sieht das wie folgt aus:

Eindimensionales Feld

A1(0)
A1(1)
A1(2)
A1(3)
A1(4)
A1(5)

Zweidimensionales Feld

Reihen \ Spalten				
	A2(0,0)	A2(0,1)	A2(0,2)	A2(0,3)
	A2(1,0)	A2(1,1)	A2(1,2)	A2(1,3)
	A2(2,0)	A2(2,1)	A2(2,2)	A2(2,3)

Das Element in der zweiten Reihe, zweiten Spalte wird A2(1,1) genannt.

Ein Beispiel für ein zweidimensionales Feld in Programmform kann so aussehen:

```
10 DIM FELD(2,3)
20 I=1
30 FOR REIHE=0 TO 2
40   FOR SPALTE=0 TO 3
50     FELD(REIHE,SPALTE)=I
60     I=I+2
70   NEXT SPALTE
80 NEXT REIHE
90 FOR REIHE=0 TO 2
100   FOR SPALTE=0 TO3
110     PRINT FELD(REIHE,SPALTE);
120   NEXT SPALTE
130   PRINT
```

```
140 NEXT REIHE

RUN

 1   3   5   7
 9   11  13  15
 17  19  21  23

READY
```

Numerische Ausdrücke und Operatoren.

Ein numerischer Ausdruck darf eine numerische Konstante oder Variable sein. Es darf ebenfalls von Kombinationen von Konstanten und Variablen Gebrauch gemacht werden, um einen bestimmten numerischen Wert zu erhalten.

Wir unterscheiden zwischen numerischen und Stringoperatoren. In den meisten Fällen werden nur die numerischen Operatoren gebraucht; wir unterscheiden auch hier wieder zwischen folgenden Kategorien:

- Arithmetische
- Relationale
- Logische
- Funktionale
  Operatoren.

Arithmetische Operatoren (Bearbeitungen).

Die arithmetischen Bearbeitungen sind die am meisten verwendeten, wie zum Beispiel addieren, subtrahieren, multiplizieren und so weiter.

Operator	Bearbeitung	Beispiel
^	potenzieren	A^B
*	multiplizieren	A*B
/	dividieren	A/B
+	addieren	A+B
-	subtrahieren	
	Negativzeichen	A-B
MOD	Modulo	A MOD B
DIV	Integerdivision	A DIV B

Bei einiger mathematischen Erfahrung werden Ihnen die meisten Bearbeitungen bekannt vorkommen. Vielleicht kennen Sie aber MODulo und die IntegerDIVision noch nicht.
MOD gibt Ihnen den Restwert nach einer Integerdivision. DIV gibt Ihnen das Ergebnis der Intergerdivision ohne Rest.

(MOD:Division ohne Rest; DIV:Bestimmung des Divisionsrestes)

Beispiel:

```
READY
 10 A=14 MOD 8
 20 PRINT A
```

```
RUN
 6
READY

 10 A=10 DIV 4
 20 PRINT A
RUN
 2
READY
```

Das Ergebnis der intergeren Teilung 14:8 ist 1 mit dem Rest 6, also gibt MOD hier den Wert 6. Das Ergebnis von DIV nach der integeren Teilung 10:4=2.5 ist also 2.
Sehen Sie sich für nähere Erklärungen bitte den Hauptteil 3 unter DIV und MOD an.

Relationale Operatoren.

Relationale Operatoren bringen zwei Werte miteinander in Beziehung (=Relation), sie vergleichen sie miteinander.Das Ergebnis eines solchen Vergleichs kann wahr (-1) oder unwahr (0) sein. Der weitere Verlauf eines Programms hängt oft vom Ergebnis eines Vergleichs ab. (Sehen Sie sich dazu auch die Anweisung IF im Hauptteil 3 an.)

Operator	Bearbeitung	Beispiel
=	ist gleich	A=B
<> oder ><	ist ungleich	A<>B
<	ist kleiner als	A<B
>	ist größer als	A>B
<= oder =<	ist kleiner oder gleich	A<=B
>= oder =>	ist größer oder gleich	A>=B

Das Gleichheitszeichen [=] wird auch benutzt, um einen Wert an eine Variable zu übergeben (Sehen Sie sich dazu auch die LET-Anweisung im Hauptteil 3 an).
Arithmetische und relationale Bearbeitungen dürfen auch zusammen benutzt werden, zum Beispiel so:

```
A+B<(X-1)/Y
```

Dieser Ausdruck ist wahr (-1), falls der Wert von A plus B kleiner ist, als der Wert von X-1 geteilt durch Y.

Sehen Sie sich auch dieses Beispiel an:

```
READY
 10 INPUT A
 20 IF A=500 THEN PRINT "ist gleich"
 30 IF A<>500 THEN PRINT "ist ungleich"
 40 IF A<500 THEN PRINT "die Zahl ist größer"
 50 IF A>500 THEN PRINT "die Zahl ist kleiner"
 60 GOTO 10
```

(Es geht hier um einfaches Zahlenraten.) Sehen Sie sich bitte auch die IF und THEN-Anweisung im Hauptteil 3 an.

Logische Operatoren.

Logische Bearbeitungen setzten eine bestimmte Logik voraus (logisch, gell?). Relationale Bearbeitungen gehen von einem Vergleich aus, um über das Ergebnis das Programm zu beeinflussen. Logische Operatoren werden jedoch meistens dazu verwendet, zwei oder mehr Vergleiche (Relationen) miteinander zu vergleichen, um dann einen "true"- (wahr) oder "false"- (unwahr) Wert zu erhalten.

Die logischen Bearbeitungen machen von "wahr-unwahr"-Werten Gebrauch und das Ergebnis ist wahr oder unwahr. Es ist "wahr", wenn der Wert ungleich Null ist (vergleichbar mit -1 bei einer relationalen Bearbeitung), oder "unwahr", wenn das, was herauskommt gleich Null ist.
Das Ergebnis einer logischen Verarbeitung ist ein Wert, der "wahr" ist, wenn er nicht gleich Null ist oder "unwahr", wenn er gleich Null ist. Der Wert wird durch einen "Bit-für-Bit"-Vergleich berechnet.
"WO IST DENN HIER DIE LOGIK?!?"

Die logischen Operatoren sind:

```
NOT  logische Verknüpfung
AND  logisches AND
OR   logisches OR
EXOR binäres exklusiv OR
&    binäres AND
!    binäres OR
```

(Die letzten drei nur in Turbo BASIC verfügbar.)

<u>Funktionale Operatoren.</u>

Eine funktionale Bearbeitung ist eine veränderliche Größe, die in ihrem Wert von einer anderen abhängig ist. Funktionale Operatoren sind:

```
SIN  (Sinus)
SQR  (Quadratwurzel)
LOG  (natürlicher Logarithmus)
CLOG (Logarithmus zur Basis 10)
ABS  (absoluter Wert)
ATN  (Arcus Tangens)
  usw...
```

Das vollständige Verzeichnis befindet sich in der Liste der reservierten Ausdrücke. Deren Erklärung folgt im Hauptteil 3 bei der diesbezüglichen Funktionsbeschreibung.

<u>Stringausdrücke und -verarbeitungen.</u>

Ein Stringausdruck (String: engl.:Kette) kann jede Ansammlung von Buchstaben, Zeichen und Ziffern oder Variablen enthalten. Ebenso können Strings aus Konstanten und Variablen durch Gebrauch von Verarbeitungsfunktionen zusammengestellt werden, um einen einzigen String zu schaffen.

Wir unterscheiden zwei Kategorien:

* Zusammenfügungen
* Bearbeitungen

Es sei angemerkt, daß auch relationale Bearbeitungen zusammen mit Strings benutzt werden dürfen, um zwei Strings miteinander zu vergleichen. Dies sind jedoch keine Stringverarbeitungen, denn das Ergebnis ist numerisch und nicht vom Typ String.

Zusammenfügungen.

Zwei Strings zu einem zusammenzufügen nennen wir Zusammenfügung. Im Englischen gibt es eine bessere Bezeichnung: Concatenation. Da ATARI- und Turbo BASIC das plus [+] Symbol bei Stringbearbeitungen nicht zulassen, gibt es Mittel und Wege, dies zu umgehen, und zwar mit Hilfe der Stringfunktion LEN. LEN ist im Hauptteil ausführlicher erklärt.

Beispiel:

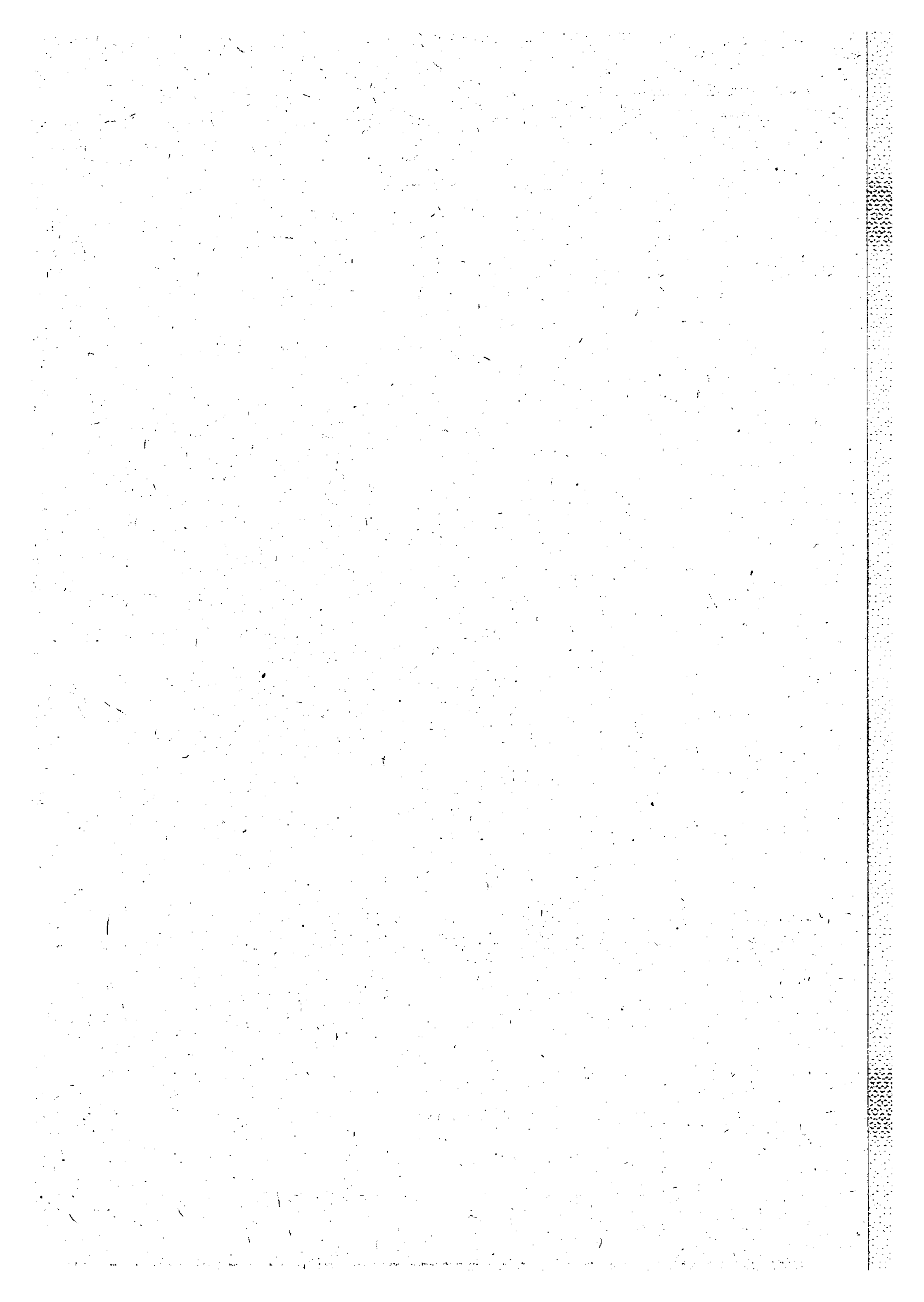
```
READY
 10 DIM A$(15),B$(15),C$(20)
 20 A$="Turbo"
 30 B$="BASIC XL 1.5"
 40 C$=A$
 50 C$(LEN(C$)+1)=B$
 60 PRINT C$
RUN
 Turbo BASIC XL 1.5
READY
```

Erst wird der Wert von A$ in den Hilfsstring C$ gesetzt (Zeile 40). Danach wird der Wert von B$ hinter den Wert von C$ gesetzt, indem die Länge von C$ bestimmt und um 1 erhöht wird.

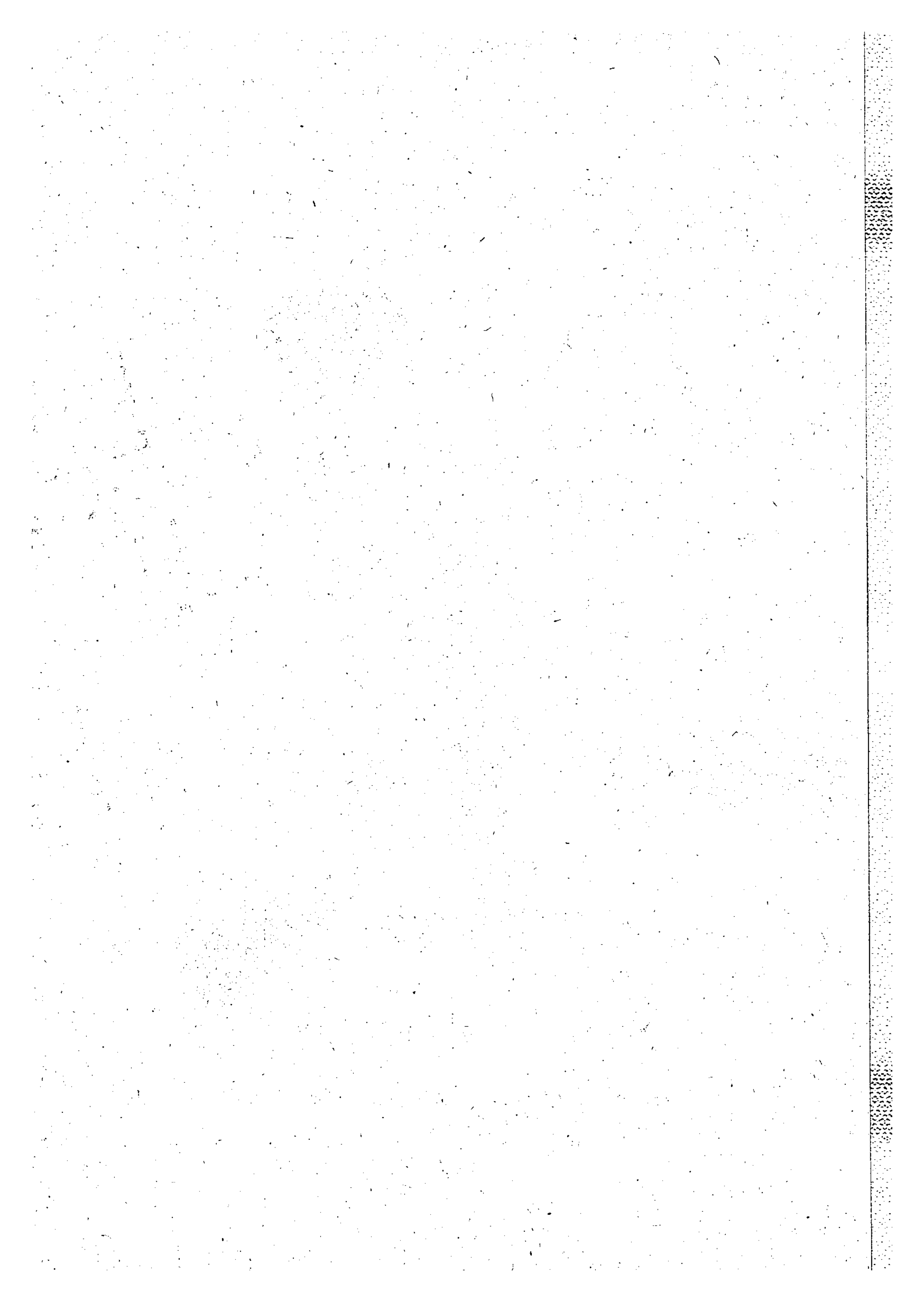
Stringbearbeitungen.

Eine Stringbearbeitung ist im Prinzip eine numerische Bearbeitung, mit dem Unterschied, daß nicht ein numerischer Wert, sondern ein String das Resultat ist.
Eine Stringfunktion kann in einem Ausdruck benutzt werden, um ein vorher bestimmtes Ergebnis zu erhalten.
BASIC besitzt verschiedene "voreingestellte" Funktionen, von denen das Ergebnis "bekannt" ist.
Einige Funktionen sind:
  ASC,CHR$,STR$,LEN,INSTR,UINSTR, usw...

Eine vollständige Beschreibung finden Sie im Hauptteil 3.




Haupt-
teil
3



ABS

Funktion

Definition:Gibt den absoluten Wert einer Zahl.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :A=ABS(x)
Anmerkung :x darf jeder numerische Wert sein.
Beispiel :READY

```
 PRINT ABS(7*(-5))
 35
READY
```

ADR
Funktion

Definition:Gibt die dezimale Speicheradresse des ersten Zeichens eines Strings.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :A=ADR(B$)
Anmerkung :B$ darf jede beliebige Stringvariable sein.
Beispiel :READY

```
 PRINT ADR("ATARI")
 65
READY
```

ASC
Stringfunktion

Definition:Gibt den ATASCII-Code des ersten Zeichens eines Strings.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :A=ASC(B$)
Anmerkung :Stringvariablen können benutzt werden. B$ kann auch z.B. ("A") oder dergleichen sein.
Beispiel :READY

```
 10 X$="TEST"
 20 PRINT ASC(X$)
 RUN
 84
READY
```

ATN
Trigonometrische Funktion

Definition:Gibt den Arcus Tangens eines Wertes im Bogenmaß oder im Winkelmaß.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :A=ATN(x)
Anmerkung :x darf jeder beliebige numerische Wert sein.
Beispiel :READY

```
 10 PI=3.141593
 20 RADIANT=ATN(1)
 30 GRAD=RADIANT*180/PI
```

```
40 PRINT RADIANT,GRAD
RUN
.7853983    45
READY
```

***B**
Systeminstruktion

Definition:Abfangen der BREAK-Taste
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :*B[Zeichen]
Anmerkung :[Zeichen] ist nicht nötig. [Zeichen] kann + oder - sein.
*B oder *B+
Nach diesem Befehl wird ein Druck auf die BREAK-Taste wie ein Fehler behandelt und kann mit TRAP umgeleitet werden. Hierdurch wird das Programm nicht unterbrochen oder nur auf eine gewollte Art.

*B-
Hebt den oben beschriebenen Zustand wieder auf. Nach RUN wird automatisch *B- ausgeführt (der Default-Wert).

BGET
I/O-Instruktion

Definition:Liest ein Byte aus einem spezifizierten Gerät und schreibt es in einen bestimmten Speicherblock.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :BGET #n,adr,len
Anmerkung :# ist ein verpflichtendes Zeichen
n ist die Kanalnummer (1 bis 7) und verweist auf eine mit OPEN geöffnete Datei.
adr ist die erste Speicheradresse, in die das erste zu lesende Byte eingesetzt werden soll.
len ist die Blockgröße der zu POKEnden Adressen.
Die BGET-Instruktion wird auch mit "Blocklesen" umschrieben.
Beispiel :BGET #3,88,7680
über Kanal 3 die Bytes lesen und diese in Adresse 88 bis 88+7680-1 POKEn.

Dasselbe sieht in ATARI BASIC so aus:

```
FOR I=0 TO len -1
  GET #n,byte
  POKE adr+I,byte
NEXT I
```

Wenn Sie für n, adr und len die gleichen Werte wie oben einsetzen, ist das Ergebnis dasselbe.

Diese Instruktion kann Daten mit maximaler Geschwindigkeit in den Speicher lesen. Dasselbe gilt für BPUT beim Schreiben.
Um z.B. den Bildschirminhalt der Grafikbetriebsstufen 9, 10, 11, oder 15 auf Diskette zu speichern, kann folgendes Beispiel benutzt werden:

```
OPEN #1,8,0,"D:BILD.PIC":BPUT #1,DPEEK(88),7680:CLOSE #1
```

Und um ihn wieder zu lesen:

OPEN #1,4,0,"D:BILD.PIC":BGET #1,DPEEK(88),7680:CLOSE #1
Die Zahl 7680 muß an die jeweiligen Bildpunkte jeder Grafikstufe angepaßt werden, da sonst Speicheradressen, die nicht zum Bildschirm gehören mitabgespeichert oder mitgeladen werden.

BLOAD
Befehl

Definition: Lädt eine binäre Datei in den Speicher.

Version	ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax : BLOAD "D[n]:dateinam.anh"

Anmerkung : Dasselbe wie die Wahl L im DOS-Menü mit angehängtem /N nach der Dateispezifikation.
[n] ist nicht zwingend und gibt die Laufwerksnummer an.

Beispiel : BLOAD "D:COMPILER.COM"

BPUT
I/O-Instruktion

Definition: Sendet ein Byte aus einem bestimmten Speicherblock an ein spezifiziertes Gerät.

Version	ATARI BASIC	Turbo BASIC	Compiler
	***	***	***

Syntax : BPUT #n,adr,len

Anmerkung : # ist ein verpflichtendes Zeichen (der IOCB-Anzeiger).
n ist die Kanalnummer (bis 7), die an eine mit OPEN geöffnete Datei verweist.
adr ist die Startadresse, von der ab geschrieben werden soll.
len ist die Anzahl der zu schreibenden Adressen ab adr.

Diese Instruktion bewirkt dasselbe wie:

```
FOR I=0 TO LEN-1
  PUT #n,PEEK(adr+I)
NEXT I
```

Sehen Sie dazu auch unter BGET, PUT und GET nach.

BRUN
Systeminstruktion

Definition: Lädt eine binäre Datei und führt diese aus, falls eine RUN-Adresse vorhanden ist.

Version	ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax : BRUN "D[n]:dateinam.anh"

Anmerkung : n ist die Laufwerksnummer und nicht verpflichtend, wenn Laufwerk 1 angesprochen werden soll (da 1 der voreingestellte (Default-) Wert ist).
dateinam.anh ist die zu ladende und startende Datei.
Diese Instruktion bewirkt dasselbe, wie die Wahl L im DOS-Menü, wenn nach der Angabe des Dateinamen nicht /N folgt.

BYE
Systeminstruktion

Definition:Verlässt BASIC und führt in den SELF TEST-Modus.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :BYE oder B.
Anmerkung :Im SELF TEST (Selbsttest)-Modus können alle Tasten, Tonkanäle und Speicherbereiche (RAM und ROM) getestet werden.

CHR$
Stringfunktion

Definition:Gibt das durch den angesprochenen ATASCII-Wert vertretene Zeichen.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :A$=CHR$(n)
Anmerkung :Reziproke Funktion von ASC.
Siehe auch PUT.
n muß einen Wert zwischen 0 und 255 haben.
Beispiel :READY

```
 PRINT CHR$(66)
  B
READY
```

oder

```
 10 FOR I=0 TO 255
 20   PRINT I;" ";CHR$(I)
 30 NEXT I
```

CIRCLE
Anweisung

Definition:Zeichnet einen Kreis um den Punkt (x,y) mit dem Radius xr.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :CIRCLE x,y,xr[,yr]
Anmerkung :Bei Angabe von zwei unterschiedlichen Radien xr und yr entstehen Ellipsen.
Beispiel :READY

```
 10 GRAPHICS 8+16
 20 CIRCLE 40,50,20
 30 GOTO 30
RUN
```

Sehen Sie sich bitte auch das Beispiel unter FILLTO an.

CLOAD
I/O-Instruktion

Definition:Lädt ein Programm von Kassette in den Speicher.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :CLOAD
Anmerkung :Nur bei Verwendung eines Kassettenlaufwerkes.
Nach Aufruf von CLOAD gibt der Rechner einen Signalton aus, der anzeigt, daß die Wiedergabetaste am Kassettenlaufwerk gedrückt werden soll. Danach muß die RETURN-Taste am Rechner betätigt werden.

CLOG
Arithmetische Funktion

Definition:Gibt den Logarithmus zur Basis 10.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :A=CLOG(x)
Anmerkung :x muß ein numerischer Ausdruck grösser als 0 sein.
Beispiel :Das erste Beispiel berechnet den Logarithmus von 45:7:

```
READY
 PRINT CLOG(45/7)
  1.860752
READY
```

Das zweite Beispiel berechnet die Logarithmen von e und e^2:

```
READY
E=2.718282
READY
PRINT CLOG(E)
 1
PRINT CLOG(E*E)
 2
READY
```

CLOSE
I/O-Instruktion

Definition:Schließt ein Gerät nach einer Eingabe/Ausgabe-Operation und gibt die IOCB-Nummer frei.
Version :ATARI BASIC Turbo BASIC Compiler
*** 1 *** 1,2 ***
Syntax :1. CLOSE #n
2. CLOSE oder CL.
Anmerkung :Darf auch benutzt werden, wenn kein Gerät geöffnet wurde. Die IOCB-Nummern sind 1 bis 7.
Siehe auch OPEN.

In ATARI BASIC darf nur die Form CLOSE #n benutzt werden.
n muß eine IOCB-Nummer von 1 bis 7 sein.

In Turbo BASIC XL 1.5 kann die oben genannte Form als auch CLOSE benutzt werden. Dies ist die verkürzte Form für:
FOR I=1 TO 7:CLOSE #I:NEXT I

CLR
Bearbeitungsinstruktion

Definition:Setzt alle numerischen Variablen auf Null. Setzt gleichfalls alle dimensionierten Felder und Strings auf Null.

Version	:ATARI BASIC	Turbo BASIC	Compiler
	***	***	***

Syntax :CLR
Anmerkung :Nachdem CLR in einem Programm ausgeführt worden ist, müssen alle Werte wieder neu dimensioniert werden.
Beispiel :

```
READY
 A=100
READY
 PRINT A
  100
READY
 CLR
READY
 PRINT A
  0
READY
```

oder

```
READY
 10 A=100:B=10
 20 C=A*B
 30 PRINT A,B,C
 40 CLR
 50 PRINT A,B,C
RUN
  100     10     1000
  0       0      0
READY
```

Die Werte, die anfänglich in den Zeilen 10 und 20 an A, B und C übergeben werden, werden durch die CLR Instruktion in Zeile 40 wieder "zunichte" gemacht.

CLS
Grafikinstruktion

Definition:Löscht den Bildschirm

Version	:ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax :CLS [#6]
Anmerkung :#6 ist nicht verpflichtend und wird nur bei einem Grafikbildschirm benutzt.
CLS hat die gleiche Wirkung wie PRINT CHR$(125).
Nach CLS wird der Bildschirm gelöscht und die Schreibmarke (Cursor) wird in die HOME-Position gebracht (Links oben, Koordinaten 0,0).

COLOR
Grafikinstruktion

Definition:Wählt eine Farbe aus dem Farbenregister in den Grafikbetriebs-
stufen 3 bis 11.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :COLOR n oder C.n
Anmerkung :Wird oftmals benutzt, um einem zu PLOTtenden Punkt einen bestimmten
Farbwert zuzuweisen.
Beispiel :READY

```
10 GRAPHICS 3+16
20 COLOR 2
30 PLOT 20,10:DRAWTO 20,20
40 GOTO 40
```

COM
Bearbeitungsinstruktion

Definition:Reserviert Speicherplatz für Strings und numerische Folgen.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :COM var(subscript)[,var(subscript)]...
Anmerkung :Ist gleich DIM.
Beispiel :Siehe DIM.

CONT
Systeminstruktion

Definition:Setzt die Ausführung eines Programms fort, nachdem die BREAK-Taste
gedrückt wurde.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :CONT
Anmerkung :Fortsetzung eines Programms nach STOP oder END ist ebenfalls möglich. Instruktionen, die in der Zeile stehen, während der BREAK gedrückt wurde oder in denen STOP oder END steht, werden nach einem CONT nicht mehr ausgeführt. Das Programm wird mit der darauffolgenden Zeile fortgesetzt.

COS
Trigonometrische Funktion

Definition:Gibt den Cosinus eines Wertes.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :A=COS(x)
Anmerkung :x ist der Wert, dessen Cosinus berechnet wird.
Beispiel :READY

```
10 PI=3.141593
20 PRINT COS(PI)
30 GRAD=180
40 RADIANT=GRAD*PI/180
50 PRINT COS(RADIANT)
RUN
```

```
 -1
 -1
READY
```

Dieses Beispiel zeigt:
1) den Cosinus von PI Bogenmass= -1
2) setzt dann das Winkelmaß in Bogenmaß um und berechnet den Cosinus von 180 Grad, welcher ebenfalls gleich -1 ist. (180 Grad=PI Bogenmaß)

CSAVE
Systeminstruktion

Definition:Speichert ein BASIC-Programm im Speicher auf Kassette ab.
Version :ATARI BASIC Turbo BASIC Compiler

ATARI BASIC	Turbo BASIC	Compiler
***	***	***

Syntax :CSAVE oder CS.
Anmerkung :Nachdem die Instruktion eingegeben und abgeschlossen wurde, meldet sich der Rechner mit 2 Signaltönen. Nun müssen am Kassettenlaufwerk die Aufnahme- und Wiedergabetaste zugleich gedrückt werden, um danach am Rechner die RETURN-Taste zu betätigen.

DATA
Bearbeitungsinstruktion

Definition:Gibt an, daß die Auflistung nach Data Informationen enthält, die mit READ gelesen werden können.
Version :ATARI BASIC Turbo BASIC Compiler

ATARI BASIC	Turbo BASIC	Compiler
***	***	***

Syntax :DATA konstante[,konstante]... oder D. konstante[,konstante]...
Anmerkung :Die Informationen nach einer Data-Anweisung können sowohl numerische als auch String-Konstanten enthalten.
Die Data-Anweisungen werden nicht vom Programm ausgeführt und dürfen deshalb an einem beliebigen Platz im Programm untergebracht werden.
Die READ-Anweisung zeigt auf die jeweils nächste zu lesene Data-Konstante.
Mit RESTORE [zeilennummer] kann zu einem willkürlichen Zeitpunkt zu einer anderen Data-Zeile gesprungen werden. Diese darf sowohl vor als auch nach der letzten gelesenen Data-Zeile liegen.
Beispiel :Siehe READ-Instruktion.

DEC
Stringfuntion

Definition:Gibt den dezimalen Wert eines hexadezimalen Strings.
Version :ATARI BASIC Turbo BASIC Compiler

ATARI BASIC	Turbo BASIC	Compiler
	***	***

Syntax :v=DEC(H$)
Anmerkung :H$ ist ein hexadezimaler Wert. Falls H$ länger als 4 Stellen ist, so gelten nur die letzten 4.
DEC ist vergleichbar mit VAL und ist das Gegenstück von HEX$.

Beispiel :READY

```
  5 DIM H$(50)
 10 FOR N=50 TO 60
 20   ? HEX$(N);" = ";N
 30   H$(LEN(H$)+1)=HEX$(N)
 40 NEXT N
 50 ? H$
 60 FOR X=1 TO LEN(H$) STEP 2
 70   ? DEC(H$(X,X+1));" = ";H$(X,X+1)
 80 NEXT X
RUN
```

Auf dem Bildschirm wird nach RUN das folgende Ergebnis stehen:

```
32=50
33=51
34=52
35=53
36=54
37=55
38=56
39=57
3A=58
3B=59
3C=60
32333435363738393A3B3C
50=32
51=33
52=34
53=35
54=36
55=37
56=38
57=39
58=3A
59=3B
60=3C
READY
```

DEG
Trigonometrische Funktion

Definition:Nach dem DEG-Befehl werden alle folgenden trigonometrischen Funktionen im Winkelmaß verarbeitet.

Version	:ATARI BASIC	Turbo BASIC	Compiler
	***	***	***

Syntax :DEG

DEL
Systeminstruktion

Definition:Löscht Programmzeilen aus dem Speicher.

Version	:ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax :DEL von,bis

Anmerkung :von ist die erste Zeilennummer, von der ab gelöscht werden soll.
bis ist die letzte Zeilennummer, bis zu der gelöscht werden soll.
Es sei angemerkt, daß DEL endgültige Wirkung zeigt. Es gibt keine Möglichkeit, aus dem Speicher gelöschte Programmzeilen wieder zurückzuholen, es sei denn, sie befinden sich noch auf dem Bildschirm. Um solche zurückzuholen, bewegen Sie einfach die Schreibmarke in diese Zeile und betätigen die RETURN-Taste.

Beispiel :Dieses Beispiel löscht nur 1 Zeile, nämlich Zeile 100:

```
DEL 100,100
```

Das folgende Beispiel löscht die Zeilen 30 bis einschließlich 100:

```
DEL 30,100
```

DELETE
I/O-Instruktion

Definiton:Löscht eine Datei von der Diskette.

Version	:ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax :DELETE "D[n]:dateinam.anh"

Anmerkung:n ist die Laufwerksnummer, ihre Angabe ist nicht zwingend, wenn Laufwerk 1 gemeint ist.
dateinam.anh ist der Name der zu löschenden Datei.
Die Benutzung von Wildcards (Platzhalter * und ?) ist erlaubt.
DELETE entspricht dem XIO-Befehl 33.

Beispiel :DELETE "D:*.BAS"
löscht in Laufwerk 1 alle Dateien mit dem Anhang BAS.

DIM
Bearbeitungsinstruktion

Definition:Reserviert Speicherplatz für numerische Folgen und Strings.

Version	:ATARI BASIC	Turbo BASIC	Compiler
	***	***	***

Syntax :DIM var(subscript)[,var(subscript)]...

Anmerkung :Jedes Stringzeichen braucht ein Byte;
jede Unterteilung in einer numerischen Folge braucht sechs Byte Speicherplatz.

Beispiel :DIM A$(10) :Ein String mit höchstens 10 Zeichen.
DIM X(10) :Eine numerische Folge, die die Elemente 0 bis 10 (also insgesamt 11) beinhaltet.
DIM Y(20,20) :Eine zweidimensionale Folge
DIM A$(10),X(10) :Es können verschiedene Variablen und Variablentypen durch ein Komma getrennt nach einer DIM-instruktion dimensioniert werden.

In Turbo BASIC XL 1.5 wird eine dimensionierte Folge gleich mit Null gefüllt. Es ist also nicht mehr notwendig, jedem Element in einem Feld einzeln eine Null zuzuweisen.
DIM A(100) bedeutet also in Turbo BASIC:

```
DIM A(100): FOR I=0 TO 100: A(I)=0: NEXT I
```

DIR
I/O-Instruktion

Definition:Zeigt das Inhaltsverzeichnis einer Diskette auf dem Bildschirm an.
Version :ATARI BASIC Turbo BASIC Compiler

ATARI BASIC	Turbo BASIC	Compiler
	***	***

Syntax :DIR ["Dn:*.*"]
Anmerkung :DIR zeigt alle Dateien der Diskette im Laufwerk 1 (1 ist der voreingestellte Wert).

Falls ein anderes Laufwerk (beispielsweise D2) gewünscht wird, muß es folgendermaßen angesprochen werden:

DIR "D2:*.*"

Platzhalter (Wildcards) dürfen auch benutzt werden, um ganz bestimmte Dateien anzeigen zu lassen. Beispielsweise alle Dateien mit der Endung BAS in Laufwerk 1:

DIR "D:*.BAS"

Diese Instruktion läßt auch Stringvariablen als Dateispezifikation zu.

DIV
Arithmetische Funktion

Definition:Division ohne Rest.
Version :ATARI BASIC Turbo BASIC Compiler

ATARI BASIC	Turbo BASIC	Compiler
	***	***

Syntax :v=a DIV b
Anmerkung :a und b sind numerische Ausdrücke.
a DIV b ergibt stets einen ganzen Wert.
Entspricht:

TRUNC(a/b)

Beispiel :? 10.25 DIV 3.33 ergibt 3
? 10 DIV 3 ergibt 3
? 3 DIV 10 ergibt 0

DO und LOOP
Programminstruktion

Definition:Eine Endlosschleife. Wiederholt stets das gleiche.
Version :ATARI BASIC Turbo BASIC Compiler

ATARI BASIC	Turbo BASIC	Compiler
	***	***

Syntax :DO: ...[Anweisung(en)]... :LOOP
Anmerkung :Die Anweisung zwischen DO und LOOP wird endlos wiederholt. Diese Schleife läßt sich normalerweise nicht verlassen. Dazu wird EXIT benötigt (siehe EXIT).
Beispiel :READY

```
10 CLS
20 DO
30   PRINT "    Hauptmenü"
40   PRINT
50   PRINT "  1 Bild laden"
```

```
60   PRINT "  2 Programm beenden"
70   PRINT "  Ihre Wahl bitte..."
80   REPEAT
90     GET TASTE
100   UNTIL TASTE>48 OR TASTE<51
110    IF TASTE=49 EXEC BILD_LADEN
120    IF TASTE=50:EXIT
130 LOOP
140 END
.
.
.
200 PROC BILD_LADEN
.
.
.
299 ENDPROC
```

Das obenstehende stellt ein strukturiertes Auswahlmenü dar. Selbstverständlich können darin noch mehr Programmteile in einer PROCedure (Unterprogramm) aufgenommen werden.

DOS
Systeminstruktion

Definition:Ruft das DOS-Menü von Diskette oder aus der RAMDisk auf.

Version	:ATARI BASIC	Turbo BASIC	Compiler
	***	***	***

Syntax :DOS

Anmerkung :Nur verwendbar im Zusammenhang mit einem Diskettenlaufwerk. DOS=Disk Operating System (engl.:Diskettenbetriebs-System). Es gibt verschiedene DOS-Versionen für ATARI Computer, zum Beispiel DOS 2.0, DOS 2.5, DOS 3.0 oder DOS 4.0. DOS 2.0 und 2.5 sind die meist verbreiteten Versionen, denn sie sind leicht zu bedienen und bräuchen relativ wenig Speicherplatz.

DPEEK
Spezielle Funktion

Definition:Liest zwei Bytes aus zwei Speicheradressen. Doppel-Byte-Peek.

Version	:ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax :A=DPEEK(adr)

Anmerkung :adr muß ein Integerwert der zu lesenden Speicheradresse zwischen 0 und 65535 sein. Der Wert, der an A übergeben wird, ist der berechnete Wert von:

```
PEEK(adr)+256*PEEK(adr+1)
```

DPOKE ist das Gegenteil von DPEEK (siehe dort). DPEEK hat keinen Einfluß auf die zu lesende Speicheradresse im Gegensatz zu DPOKE.

Beispiel :

```
READY
 A=DPEEK(560): PRINT A
 48160
READY
```

DPOKE
Bearbeitungsinstruktion

Definition:Schreibt zwei Bytes in zwei Speicheradressen. Doppel-Byte-POKE.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :DPOKE adr,word
Anmerkung :adr muß ein Integerwert für eine Speicheradresse zwischen 0 und 65535 sein, in die word geschrieben wird. Ebenfalls wird ein Wert in adr+1 geschrieben.
word ist ein Wert oder numerischer Ausdruck, der in die angegebene Adresse adr geschrieben wird.
Beispiel :DPOKE 560,100
bewirkt folgendes:

```
POKE 560,100-256*INT(100/256): POKE 561,INT(100/256)
```

oder allgemeiner:

```
POKE adr,word-256*INT(word/256): POKE adr+1,INT(word/256)
```

DPOKE spart nicht nur Speicherplatz, sondern auch die Arbeit, unübersichtliche Befehlsfogen einzutippen.
Sehen Sie sich dazu auch DPEEK, POKE und PEEK an.

DRAWTO
Grafikinstruktion

Definition:Zeichnet eine Gerade zwischen zwei Punkten.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :DRAWTO x,y oder DR. x,y
Anmerkung :x und y sind die Koordinaten auf der x- und y-Achse.
x ist die Spalte und y die Zeile.
Der Anfangspunkt der Linie, die gezeichnet wird, ist die jeweils letzte Position des Cursors. Der Endpunkt wird mit x und y angegeben.
Beispiel :READY

```
 10 GRAPHICS 11
 20 FOR Y=0 TO 191
 30   C=Y/16
 40   X=SQR(191-Y)*2
 50   PLOT X,Y
 60   DRAWTO X+7,Y
 70   PLOT 36,Y
 80   DRAWTO 43,Y
 90   PLOT 72-X,Y
 100   DRAWTO 79-X,Y
 110 NEXT Y
 120 GOTO 120
```

DSOUND
Toninstruktion

Definition:Gibt einen Ton aus dem Lautsprecher des Fernsehgerätes oder des Monitors aus.
Version :ATARI BASIC Turbo BASIC Compiler

ATARI BASIC	Turbo BASIC	Compiler
	***	***

Syntax :DSOUND kanal,frequenz,verzerrung,lautstärke
DSOUND
Anmerkung :ähnlich dem normalen SOUND-Befehl.
ATARI Computer können zwei normale Stimmen zu einer zusammenfassen. Die Frequenzauflösung beträgt dann 16 (0...65535) statt 8 (0...255) Bit. Die resultierende Frequenz berechnet sich in Hertz zu: 1789790/(2*frequenz+14).
Sehen Sie sich bitte auch die Erläuterungen zu SOUND an.

DUMP
Systeminstruktion

Definition:Zeigt die Variablen und Prozeduren in einem Programm
Version :ATARI BASIC Turbo BASIC Compiler

ATARI BASIC	Turbo BASIC	Compiler
	***	

Syntax :DUMP [dateispezifikation]
Anmerkung :Die Dateispezifikation ist nicht zwingend vorgeschrieben. Sie gibt das Gerät an, auf das die Liste der Variablen und Prozeduren geschrieben (geDUMPt) werden soll. Der Drucker (P:) könnte zum Beispiel als Ausgabegerät dienen. Der voreingestellte Wert bei der Verwendung von DUMP ist der Bildschirm.
Beispiel :DUMP "P:"
Vergessen Sie nicht die Anführungszeichen und den Doppelpunkt.

Eine DUMP-Liste kann wie folgt aussehen:

```
A = 100      :numerische Variable
B(10,1       :ein Feld; DIM B(9) oder DIM B(9,0)
C(0,0        :undimensioniertes Feld (Array)
              Bei Feldern werden beide möglichen Dimensionen stets
              um 1 erhöht angezeigt
D(10,10      :DIM D(9,9)
E$ 10,20     :Stringvariable, deren Länge 10 ist; DIM E$(20)
F$ 0,0       :nicht dimensionierte Stringvariable
G$ 0,10      :Länge von G$ ist 0; DIM G$(10)
H PROC 100   :Prozedur H beginnt in Zeile 100
I # 120      :Dieses Label beginnt in Zeile 120
J ?          :Nicht benannte Zeile oder Prozedur
```

Die Reihenfolge der in der DUMP-Liste aufgeführten Elemente ist identisch mit der Reihenfolge der Elemente in der Variablentafel.

END
Anweisung

Definition:Beendet ein Programm, schließt alle geöffneten Kanäle und stoppt die Tonausgabe.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :END
Anmerkung :Die END-Anweisung darf an jedem beliebigen Platz im Programm stehen.
END unterscheidet sich auf drei Arten von STOP:
1. END veranlaßt den Rechner zu keiner Meldung (STOPPED IN LINE).
2. END schließt alle geöffneten Kanäle.
3. END beendet die Tonausgabe.
Beispiel :500 IF WAHL=0 THEN END
Dieses Beispiel beendet das Programm, wenn die Variable WAHL gleich Null ist. Andernfalls wird das Programm mit der nächsten Zeile fortgesetzt.

ENDPROC
Programminstruktion

Definition:Bezeichnet das Ende einer Prozedur (Unterprogramm).
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :ENDPROC
Anmerkung :Diese Instruktion ist mit der RETURN-Instruktion bei einem Unterprogramm mit GOSUB vergleichbar.
Schlagen Sie für weitere Erläuterungen und einem Beispiel bitte unter EXEC nach.

ENTER
Systeminstruktion

Definition:Fügt die Programmzeilen einer ATASCII-Datei in ein Programm ein, das sich bereits im Speicher befindet.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :ENTER "gerät[nummer]:dateinam.anh"
oder
E. "gerät[nummer]:dateinam.anh"
Anmerkung :gerät bezeichnet das Quellgerät, zum Beispiel C für das Kassettenlaufwerk und D für das Diskettenlaufwerk.
nummer ist die Laufwerksnummer (optional).
: ist ein verpflichtendes Zeichen.
dateinam.anh gibt den Dateinamen und den Anhang an. Diese Datei muß zuvor mit dem LIST-Befehl auf das Medium geschrieben worden sein.
Seien Sie bitte vorsichtig, wenn Sie von ENTER Gebrauch machen, da Programmzeilen, die dieselbe Zeilennummer wie eine bereits im Speicher befindliche haben, diese überschreiben.
Beispiel :ENTER "C:"

ENTER "D2:TEIL3.LST"

ERR und ERL
Variablen

Definition:Gibt die Fehlernummer und die Nummer der Zeile, in der ein Fehler aufgetreten ist.

Version :ATARI BASIC , Turbo BASIC Compiler
 *** ***

Syntax :ERR=fehlernummer
ERL=fehlerzeile

Anmerkung :ERR ist die verkürzte Schreibweise für DPEEK(195); die Adresse, in der die Fehlernummer gespeichert wird.
ERL ist die verkürzte Form für PEEK(186)+256*PEEK(187) oder einfach DPEEK(186). Dies ist die Adresse, in der die Zeilennummer gespeichert wird, wenn ein Fehler festgestellt wird.
ERR und ERL lassen sich sehr gut in TRAP-Routinen benutzen.

Beispiel :READY

```
10 TRAP #FEHLER
20 LPRINT "Dies wird auf dem Drucker ausgegeben."
30 END
100 #FEHLER
110 IF ERR=138 THEN PRINT "Der Drucker ist nicht ONLINE."
120 PRINT "Der Fehler trat in Zeile ";ERL;" auf."
130 RUN
```

Falls ein Fehler auftritt (im obenstehenden Beispiel, wenn der Drucker nicht auf ONLINE gestellt ist), wird das Programm nicht unterbrochen, sondern durch TRAP zu der mit FEHLER benannten Zeile 100 geleitet. Hier wird eine Fehlermeldung angezeigt und das Programm erneut gestartet.

EXEC
Programminstruktion

Definition:Springt in eine Prozedur (Unterprogramm), die einen bestimmten Namen trägt.

Version :ATARI BASIC Turbo BASIC Compiler
 *** ***

Syntax :EXEC name

Anmerkung :name ist der Name der Prozedur

EXEC ist vergleichbar mit GOSUB. Die Anweisungen GOSUB und RETURN können ein Unterprogramm jedoch nur über eine Zeilennummer aufrufen, während EXEC name dies mit einem Namen bewerkstelligt, den man frei wählen und gut behalten kann. Wenn sich viele Unterprogramme (Subroutinen) in einem Programm befinden, verliert man bei ausschließlicher Verwendung von Zeilennummern schnell den Überblick. Zeilennummern haben außerdem den Nachteil, daß bei einer Neuordnung des Programms alle GOSUB-Anweisungen geändert werden müssen.
Zudem braucht eine EXEC-PROC-ENDPROC-Konstruktion weniger Speicherplatz und arbeitet schneller als GOSUB-RETURN.
Die Prozedurnamen werden genauso wie die Variablennamen verwaltet. Jeder Name braucht 8 Bytes plus 1 Byte für jedes Zeichen. Die Zeilennummer von einem GOSUB verbraucht dagegen nur 7 Bytes.

Turbo BASIC ist zwar kompatibel zu ATARI BASIC, es können aber 25 statt nur 128 Variablen oder Prozedurnamen benutzt werden.

Ab der 129. Variable kostet jede Anwendung jedoch gut zwei anstatt eines Bytes Speicherplatz.

Beispiel :READY

```
 10 CLS
 20 EXEC INITIALISIERUNG
 30 EXEC ANZEIGE
 40 DO
  .
  .
  .            Hauptprogramm
  .
  .
 100 LOOP
 110 END
 199 ---------------------------------
 200 PROC INITIALISIERUNG
 210   DIM A$(100),B$(100),FELD(100)
 220   AUS=40000
 230 ENDPROC
 299 ---------------------------------
 300 PROC ANZEIGE
 310   DL=DPEEK(560)
 320   POKE DL+3,70:POKE DL+6,7
 330   PRINT #6;"TURBO BASIC MANUAL"
 340   PRINT #6;"    ABBUC e.V.    "
 350 ENDPROC
```

In Zeile 10 wird der Bildschirm gelöscht. Zeilen 20 und 30 rufen die Prozeduren INITIALISIERUNG und ANZEIGE auf. In den Zeilen 40 bis 100 folgt die Ausführung des Hauptprogramms.
In den Zeilen 200 bis 230 wird die INITIALISIERUNG abgearbeitet, nachdem diese mit EXEC (execute, engl.:ausführen) aufgerufen wurde.
In den Zeilen 300 bis 350 befindet sich die Prozedur ANZEIGE.
Auch diese wird noch vor dem Hauptprogramm mit EXEC aufgerufen.

Die obige Konstruktion ist ein Beispiel für strukturierten Programmaufbau.
Sehen Sie sich bitte auch ON und EXEC an.

EXIT
Programminstruktion

Definition:Verläßt eine Schleife.

Version :ATARI BASIC Turbo BASIC Compiler
                     ***   ***

Syntax :EXIT

Anmerkung :EXIT verläßt eine Schleife und springt zum Ende der betreffenden Schleife. EXIT funktioniert bei Schleifen, die mit DO-LOOP, REPEAT-UNTIL, WHILE-WEND und FOR-NEXT gebildet werden.
Man kann dieses Instruktion als eine Art "Notausgang" verstehen; bei DO-LOOP ist EXIT sogar der einzige Ausweg, der beim strukturierten Programmieren erlaubt ist.
Eine Schleife kann auch mit POP: GOTO zeilennummer verlassen werden. Dies sollten Sie jedoch meistens vermeiden, da Programme dadurch nicht gerade an Übersichtlichkeit gewinnen.

Beispiel :Siehe unter DO-LOOP.

EXP
Arithmetische Funktion

Definition:Berechnet den Exponent einer numerischen Variable.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :A=EXP(x)
Anmerkung :x darf jeder beliebige numerische Ausdruck sein.
Gibt den Wert der Eulerschen Zahl e (ungefähr 2.718) hoch x.
In einigen Fällen ist EXP nur auf 6 Stellen genau. EXP ist reziprok zur LOG-Funktion.
Beispiel :

```
READY
 10 X=2
 20 PRINT EXP(X-1)
RUN
 2.718282
```

***F**
Spezielle Funktion

Definition:Schleifenprüfer
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :*F[zeichen]
Anmerkung :zeichen ist nicht zwingend vorgeschrieben und kann + oder - sein.

*F oder *F+

Nach dieser Anweisung wird bei einer FOR...NEXT-Schleife erst geprüft, ob der Zähler den Endwert bereits erreicht hat, bevor die eigentliche Schleife ausgeführt wird.
Ist dies der Fall, dann wird die Schleife bis zum NEXT überschlagen und es folgt keine Ausführung der Schleife.

*F-

stellt den normalen Zustand wieder her. FOR...NEXT-Schleifen werden dann mindestens einmal durchlaufen. Auch bei RUN wird vom System automatisch ein *F- ausgeführt.
Beispiel :

```
 10 *F+
 20 FOR X=3 TO 1
 30   PRINT X
 40 NEXT X
 50 END
```

Der Anfangswert der Schleife ist 3; der Schleifenzähler wird darauf um 1 erhöht.
Dieser Wert wird mit dem Endwert, welcher 1 ist, verglichen. Die Schleife wird verlassen, weil die Bedingung wahr (TRUE) ist.
Nach *F oder *F+ wird erst die Variable X mit dem Anfangswert 3 geladen und dann mit der Variable X (Endwert=1) verglichen.
Die Schleife wird bis zum NEXT X überschlagen.
Es findet keine Bildschirmausgabe statt.

FCOLOR
Grafikinstruktion

Definition:Wählt eine Farbe aus dem Farbregister.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :FCOLOR n
Anmerkung :n ist der Farbwert aus dem Farbregister. Dieser muß ein Integer-
wert sein und darf die Werte 0, 1, 2 oder 3 erhalten.

FCOLOR kann nur im Zusammenhang mit FILLTO benutzt werden.
FCOLOR gibt also den Farbwert für den zu füllenden Bereich.
Im ATARI BASIC heißt dieser Befehl POKE 765,n.
Beispiel :Siehe FILLTO.

FILLTO
Grafikinstruktion

Definition:Füllt eine begrenzte Fläche mit einer ausgewählten Farbe.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :FILLTO x,y
Anmerkung :x gibt die x-Koordinate auf dem grafischen Bildschirm an (Spalte).
y gibt die y-Koordinate auf dem grafischen Bildschirm an (Zeile).

FILLTO ist die Kurzschreibweise von:
POSITION x,y: XIO 18,#6,0,0,"S:": POKE 765,farbe

Um eine Fläche einzufärben, muß diese erst einmal eingegrenzt
werden. Die Begrenzung kann wie folgt von statten gehen:
PLOT ein Punkt rechtsoben
DRAWTO irgendwo rechtsoben
DRAWTO irgendwo linksunten
FILLTO irgendwo bis linksunten
Geben Sie der Fläche mit FCOLOR (Erklärung siehe dort) eine
Farbe.
Das schwierige an FILLTO oder XIO 18 ist, daß die linke Begrenzung
immer eine gerade Linie sein muß, da sonst diese Befehle nicht
korrekt arbeiten.
Sehen Sie sich dazu bitte auch PAINT an.
Beispiel :

```
10 GRAPHICS 24
20 SETCOLOR 2,10,0
30 SETCOLOR 1,6,10: COLOR 1
40 PLOT 250,170: REM Rechtsoben
50 DRAWTO 175,10: REM Rechtsoben
60 DRAWTO 125,10: REM Linksoben
70 FILLTO 50,70: REM Linksunten
80 FCOLOR 1
90 TEXT 110,100,"TURBO BASIC"
100 CIRCLE 40,80,30
110 CIRCLE 270,80,30
120 PAINT 40,80: PAINT 270,80
130 GOTO 130
```

FOR und NEXT
Programminstruktion

Definition:Wiederholt eine Folge von Anweisungen innerhalb der Schleife in einer bestimmten Anzahl.

Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :FOR var=x TO y [STEP z]

```
.
.
NEXT var[,var]...

oder

F. var= ...(siehe oben)
.
.
N. var ...(siehe oben)
```

Anmerkung :var ist ein Integerwert, der als Zähler benutzt wird.
x ist ein numerischer Ausdruck, der den Anfangswert für var darstellt.
y ist ein numerischer Ausdruck, der den Endwert für var darstellt.
z ist ein numerischer Ausdruck, der die Schrittgröße angibt.
STEP z ist nicht zwingend vorgeschrieben. Die voreingestellte Schrittgröße ist 1.
Um den guten Verlauf eines Programmes zu gewährleisten, soll eine Schleife nur mittels der POP-Instruktion in ATARI BASIC und der POP- oder EXIT Anweisungen in Turbo BASIC verlassen werden.

Beispiel :FOR ZAEHLER=1 TO 10

```
.
.
NEXT ZAEHLER
```

Der ZAEHLER soll mit 1 beginnen und immer wieder um 1 erhöht werden, bis 10 erreicht ist.

```
FOR A=10 TO 1 STEP -2
```

A beginnt mit 10 und wird mit jedem Durchlauf um 2 erniedrigt.

```
FOR A=QIN TO P*S STEP Z
```

Berechnete Werte für x, y und/oder z sind auch erlaubt.

```
READY
 10 FOR X=0 TO 255
 20   POKE 712,X
 30   FOR WARTE=1 TO 25: NEXT WARTE
 40 NEXT X
```

Dieses kleine Programm wechselt die Farbe des Bildschirmrahmens und zeigt dabei alle Farben, die der ATARI Computer darstellen kann.

FRAC
Arithmetische Funktion

Definition:Ermittelt den Nachkommateil einer Zahl.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :v=FRAC(exp)
Anmerkung :exp stellt einen numerischen Ausdruck dar.
FRAC(exp) ist nicht immer gleich:
exp-INT(exp),
weil INT die nächstkleinere Zahl ermittelt.
Beispiel :INT(-0.3) ergibt -1 und
FRAC(-0.3) ergibt -0.3.
Siehe auch TRUNC.

FRE
Spezielle Funktion

Definition:Ermittelt den noch freien Speicherplatz.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :FRE(0)
var=FRE(0)
Anmerkung :(0) ist ein vorgeschriebener "Dummy"-Wert, das heißt, das dieser Wert keinerlei Einfluß auf die Funktion hat, aus internen Gründen aber diese Syntax eingehalten werden muß. Statt 0 kann jeder andere beliebige numerische Ausdruck benutzt werden.

Das Ergebnis dieser Funktion liefert den dezimalen Wert des noch verfügbaren RandomAccessMemory (Speicher mit wahlfreiem Zugriff).
Beispiel :READY

```
 PRINT FRE(X)
 32274 :in ATARI BASIC
 34017 :in Turbo BASIC XL 1.5
READY
```

GET
I/O-Instruktion

Definition:Liest ein einzelnes Byte aus einem spezifizierten Gerät und setzt es in die angegebene Variable ein.
Version :ATARI BASIC Turbo BASIC Compiler
*** 1 *** 1,2 ***
Syntax :1: GET #n,var
2: GET var
Anmerkung :n spricht ein bestimmtes Eingabegerät an und muß einen integren Wert von 1 bis 7 haben.
Der voreingestellte Wert für das Eingabegerät ist in Turbo BASIC die Tastatur. Sobald das Programm auf ein GET stößt, stoppt es, wartet auf einen Tastendruck und setzt danach den ATASCII-Wert der Taste in die Variable.
Beispiel :In ATARI BASIC:

```
 10 OPEN #1,4,0,"K:"
 20 GET #1,A
 30 PRINT A
 40 CLOSE #1
```

```
50 GOTO 10
```

oder das Lesen von Diskette oder Kassette:

```
10 OPEN #1,4,0,"D:DATEI.DAT"
20 GET #1,A
30 FOR I=1 TO A
40 GET #1,B
50 PRINT B
60 NEXT I
70 CLOSE #1
```

In Turbo BASIC XL 1.5:
GET TASTE
ist das gleiche wie
OPEN #1,4,0,"K:": GET #1,TASTE: CLOSE #1

```
10 GET TASTE
20 PRINT TASTE
30 GOTO 10
```

Diese kleine Beispielprogramm bewirkt genau das gleiche wie das allererste Beispiel in ATARI BASIC. Man kann also mit Recht sagen, das dies hier bedeutend kürzer ist.

Zum Arbeiten mit Datenbeständen empfiehltes sich, wenn Sie sich auch %GET und BGET ansehen.

%GET
I/O-Instruktion

Definition:Holt ein Byte aus einem spezifizierten Gerät.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :%GET #n,byte
Anmerkung :% zeigt dem Interpreter an, daß das Byte, das geholt werden soll, in komprimierter Form (6-Byte-Form) abgespeichert wurde.
Dadurch kommt eine schnellere Datenübertragung vom Speichermedium zum Rechner zustande.
ist ein verpflichtendes Zeichen.
n ist die Kanalnummer (1 bis 7), die zu einem mit OPEN geöffneten Kanal verweist.
byte gibt die zu holende Zahl an.
Beispiel :Mit dem folgenden Beispiel werden beliebige Zahlen, die mit dem Programmbeispiel unter %PUT (siehe dort) abgespeichert wurden, in den Speicher gelesen.
Dabei werden die Zahlen, in ein Feld übertragen, auf dem Bildschirm angezeigt.

```
100 CLR: DIM FELD(100)
110 OPEN #1,4,0,"D:DATEI.DAT"
120 FOR I=1 TO 100
130   %GET #1,BYTE
140   FELD(I)=BYTE
150   PRINT BYTE,CHR$(BYTE)
160 NEXT I
170 CLOSE
```

GO#
Programminstruktion

Definition:Springt zu einer Zeile mit einem angegebenen Namen (Label).
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :GO# name
Anmerkung :# ist ein verpflichtendes Zeichen.
name ist der Labelname, ähnlich dem Prozedurnamen.

GO# name bewirkt das gleiche wie GOTO zeile in ATARI BASIC. Es wird deutlich, dass ein unstrukturierter Sprungbefehl wie GOTO zeile ein Stück lesbarer wird, wenn er durch GO# name ersetzt werden kann.
Das vorgeschriebene Zeichen # nach GO teilt dem Interpreter mit, daß keine Zeilennummer, sondern ein Name folgt.
Es sei noch hinzugefügt, dass GO# name schneller als GOTO zeile ist.
Das untenstehende Beispiel ist gleichfalls eine Anwendung von TRAP und RESTORE.

Beispiel :
```
5 B=10
10 INPUT "Geben Sie eine Zahl ein ";A
20 IF A=100: GO# LABEL
30 PRINT A
40 B=B+A
50 GOTO 10
100 # LABEL
110 PRINT A,B
120 END
```

Dieses Beispiel ist nur unter Turbo BASIC lauffähig.
Siehe auch GOTO.

GOSUB und RETURN
Programminstruktion

Definition:Springt in ein Unterprogramm, führt dieses aus und springt zurück zur Anweisung nach GOSUB.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :GOSUB zeile oder GOS. zeile
.
.
RETURN oder RET.
Anmerkung :zeile ist die Zeilennummer der ersten Zeile des Unterprogramms.
Die Ausführung eines Programms im internen Speicher wird von der angegebenen Zeile an bis zur ersten RETURN-Instruktion fortgesetzt. Diese Instruktion läßt die Ausführung zu der Anweisung zurückkehren, die direkt nach dem GOSUB folgt.
zeile darf auch einen berechneten Wert haben, zum Beispiel GOSUB X+10

Beispiel :
```
READY
10 INPUT A
20 PRINT A
30 GOSUB 100
40 PRINT A,B
50 END
```

```
99 REM ----Unterprogramm Berechnung----
100 B=A+10
110 RETURN
```

Sehen Sie sich bitte auch EXEC-PROC-ENDPROC an.

GOTO
Programminstruktion

Definition:Springt zu einer angegebenen Programmzeile.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :GOTO zeilennummer oder GO TO zeilennummer
Anmerkung :zeilennummer ist eine beliebige Zeilennummer oder ein berechneter Wert, an dem die Programmausführung fortgesetzt werden soll.
Beispiel :

```
10 INPUT X
20 IF X<100 THEN GOTO 500
30 GOTO 10
500 PRINT "X ist kleiner als 100!"
510 PRINT "Um genau zu sein: X hat den Wert ";X
520 GOTO 10
```

Ein anderes Beispiel:

```
10 X=RND(0)*10+1
20 IF X<5 THEN 10
30 PRINT X
40 GOTO 10
```

Siehe auch GO# name.

HEX$
Stringfunktion

Definition:Wandelt einen Integerwert in einen hexadezimalen String um.
Version :ATARI BASIC Turbo BASIC XL Compiler
*** ***
Syntax :v$=HEX$(n)
Anmerkung :n ist ein numerischer Ausdruck zwischen 0 und 65535, der gleichfalls einen Integerwert darstellt (eine Ganzzahl). Falls n kleiner als 256 ist, wird das Ergebnis von HEX$ ein String mit 2 Stellen sein, sonst ist er vierstellig. HEX$ ist vergleichbar mit STR$.
Beispiel :

```
READY
 10 FOR N=250 TO 260
 20 PRINT HEX$(N);" = ";N
 30 NEXT N
RUN
 FA = 250
 FB = 251
 FC = 252
 FD = 253
 FE = 254
 FF = 255
 0100 = 256
 0101 = 257
```

0102 = 258
0103 = 259
0104 = 260

Sehen Sie sich in diesem Zusammenhang bitte auch DEC an.

IF und THEN
Programminstruktion

Definition:Vergleich zweier angegebener Werte nach IF. Falls der Vergleich wahr ist, wird die Anweisung nach THEN ausgeführt.

Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :IF var [vergleich][var][logischer operator][var] THEN anweisung

Anmerkung :var darf jeder numerische oder Stringausdruck sein.
vergleich kann sein:
= ist gleich
<> ist ungleich
< ist kleiner als
> ist größer als
<= ist kleiner als oder gleich
>= ist größer als oder gleich
logischer operator kann sein:
AND logisches AND
OR logisches OR
NOT logische Verneinung
anweisung kann jede beliebige Anweisung oder Folge von Anweisungen sein.

Beispiel :

```
IF X=Y THEN PRINT "X ist gleich Y."
IF X<>Y THEN PRINT "X ist ungleich Y."
IF X<Y THEN GOTO 100
IF X>Y THEN GOTO 300
IF X<=Y THEN GOSUB 1000
IF X>=Y THEN END
IF X=Y AND A=B THEN SOUND 1,121,10,10
IF X=Y OR A=B THEN GRAPHICS 0
IF X THEN PRINT "X ist ungleich Null"
```

IF ... ELSE ... ENDIF
Programminstruktion

Definition:Vergleich mehrerer Variablen, so daß Anweisung 1 oder Anweisung 2 ausgeführt werden kann.

Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :IF aexp anweisung1 [ELSE anweisung2] ENDIF

Anmerkung :Wenn die Bedingung nach IF (aexp) erfüllt ist, wird Anweisung 1 ausgeführt, sonst wird Anweisung 2 ausgeführt.
Zur Unterscheidung von aexp und anweisung steht nicht THEN, sondern ein Doppelpunkt oder das Zeilenende.
Nach ELSE darf kein Sprungbefehl GOTO benutzt werden.
ENDIF dient dazu, das Ende der Schleife oder Anweisung(en) anzugeben. Es ist nicht nötig, IF ... ELSE ... ENDIF in eine Zeile zu schreiben.

Beispiel :
```
10 INPUT "Geben Sie eine Zahl ein ";A
20 IF A<10                                Vergleich
30   PRINT "A ist < 10; A = ";A           Anweisung 1
40     ELSE
50     PRINT "A ist > 10; A = ";A         Anweisung 2
60 ENDIF
70 GOTO 10
```

INKEY$

Spezielle Funktion

Definition: Liest ein Zeichen, daß über die Tastatur eingegeben wurde, falls eine Taste gedrückt wurde.

Version	ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax : K$=INKEY$

Anmerkung : K$ ist eine Stringvariable, in die das betreffende Zeichen geschrieben wird.
Falls keine Taste gedrückt wird, enthält K$ einen Leerstring "".
So läßt sich ein Tastendruck verarbeiten, ohne den Programmablauf zu unterbrechen.

INPUT

I/O-Instruktion

Definition: Liest Daten aus einem Gerät ein und setzt sie in eine numerische oder Stringvariable.

Version	ATARI BASIC	Turbo BASIC	Compiler
	***	***	***

Syntax : ATARI BASIC und Turbo BASIC:
```
INPUT [#kanal,] var [,var] ...
```

Turbo BASIC:
```
INPUT ["text",] var [,var] ...
INPUT ["text";] var [,var] ...
```

oder als Abkürzung für INPUT nur I.

Anmerkung : Der Inhalt einer Zeile von Daten muß mit einem <RETURN>-Zeichen oder CHR$(13) abgeschlossen werden.
kanal ist ein IOCB-Kanal, der erst mit der OPEN-Instruktion geöffnet werden muß.
kanal ist eine Zahl von 1 bis 7.
var darf jede beliebige numerische oder Stringvariable sein.

Turbo BASIC XL 1.5:
Diese INPUT-Instruktion erinnert stark an das MICROSOFT BASIC.
Es ist möglich, einen Text nach INPUT zu benutzen. Wird nach dem Text ein Komma gesetzt, erscheint bei der Eingabe ein Fragezeichen.
Möchte man dieses manchmal störende Fragezeichen auf dem Bildschirm unterdrücken, muß man anstelle des Kommas ein Semikolon (;) setzen.

Beispiel :
```
INPUT A        (Fordert einen numerischen Wert und setzt ihn
                in die Variable A)
INPUT X,Y,Z    (Fordert drei Zahlen, durch Kommata getrennt und
                setzt diese in X, Y und Z.)
```

INPUT A$ (Fordert eine Zeichenfolge und setzt diese in A$, ausgenommen das <RETURN>-Zeichen; A$ muß vorab dimensioniert werden; siehe DIM)

```
 10 OPEN #1,4,0,"D:DATEI.DAT"
 20 INPUT #1,A$,A
 30 CLOSE #1
```

Dieses Beispiel holt eine Zeichenfolge aus der Datei DATEI.DAT, die sich auf einer Diskette in Laufwerk 1 befinden soll, und setzt sie in die Variablen A$ und A.

Für Turbo BASIC:

```
INPUT "Geben Sie eine Zahl ein   ",A
INPUT "Geben Sie Ihren Namen ein ";N$
INPUT "";A
```

Die Abkürzung I. darf auch hier benutzt werden.

INSTR
Stringfunktion

Definition:Sucht einen String in einem längeren String.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Synatx :v=INSTR(v$,x$[,i])
Anmerkung :INSTR sucht in dem längeren String v$ den kürzeren String x$ und setzt den Positionswert, wo x$ in v$ gefunden wurde, in die Variable v.
i (falls angegeben) ist der Anfangswert, von dem ab gesucht werden soll. Falls i nicht angegeben wird, geht der Interpreter von i=0 aus.
Beispiel :

```
READY
 10 DIM A$(40),B$(10)
 20 A$="ABBUC-JAHRESHAUPTVERSAMMLUNG-1990"
 30 B$="VERS"
 40 A=INSTR(A$,B$)
 50 PRINT A
RUN
 18
READY
```

Dieses Beispiel sucht das Wort VERS in dem viel längeren A$ und setzt die Position, ab der VERS beginnt, in die Variable A. Dann wird das Ergebnis auf dem Bildschirm angezeigt.
(Das Ergebnis ist also 18)

INT
Arithmetische Funktion

Definition:Gibt die größte ganze Zahl, die kleiner oder gleich einem angegebenen Ausdruck ist.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :v=INT(x)
Anmerkung :v ist eine numerische Variable.
x darf jeder numerische Ausdruck einer numerischen Bearbeitung sein. Ebenso dürfen in der Bearbeitung andere arithmetische Funktionen vorkommen.
Beispiel :READY

```
 10 X=2.3456
 20 A=INT(X)
 30 PRINT "A = ";A
RUN
 A = 2
READY

A=INT((RND(0)*10)+0.05)/10
```

***L**
Bearbeitungsfunktion

- Definition:Tabulatoreinstellung für die LIST-Instruktion.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :*L[zeichen]
Anmerkung :zeichen ist nicht zwingend vorgeschrieben; zeichen kann + oder - sein.

*L oder *L+

Nach dieser Anweisung wird der Tabulator (2 Leerschritte) eingeschaltet. Dies ist auch der normale Zustand nach dem Laden von Turbo BASIC. Schleifen werden 2 Leerschritte eingerückt.

*L-

Schaltet den Tabulator für die LIST-Instruktion aus, so daß, falls notwendig, mehr Platz entsteht. Dies kann auch beim Editieren von langen Programmzeilen von Nutzen sein, oder um Platz auf der Diskette zu sparen, wenn ein Programm mit LIST "D:..." abgespeichert wird.

Das Übersichtszeichen -- wird nach *L- statt dreißigfach nur noch zweifach ausgegeben

LEN
Stringfunktion

Definition:Bestimmt die Länge einer Stringvariablen und setzt diese in eine numerische Variable.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :x=LEN(A$)
x=LEN("STRING AUSDRUCK")
Anmerkung :Die Anzahl der Zeichen wird bestimmt, die sich in einer Stringvariable befinden, ungeachtet dessen, um welche Zeichen oder Leerschritte es sich handelt.

Beispiel :READY

```
 10 DIM A$(100)
 20 A$="DIES IST EIN STRING"
 30 A=LEN(A$)
 40 PRINT A
RUN
 19
READY
```

Auf die folgende Art kann die Länge eines Strings für das Ergebnis einer Berechnung wichtig sein:

```
 50 B=A*LEN(A$)
 60 PRINT B
RUN
 361
READY
```

LET
Bearbeitungsinstruktion

Definition:Weist einer numerischen oder Stringvariablen einen Wert zu.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :[LET] var[$]=["]wert["]
Anmerkung :Der Gebrauch der Instruktion LET ist nicht verpflichtend.
Beispiel :LET A=B
Der Wert von B wird A zugewiesen.

LET A$="Markus Müller"

oder
A=B
A$="Markus Müller"

LIST
Systeminstruktion

Definition:Zeigt ein Programm auf dem Bildschirm oder gibt es an ein anderes Peripheriegerät.

Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :LIST [gerät n:][dateispezifikation][,zeile][,zeile]

Anmerkung :Es gibt einen Unterschied zwischen dem LISTen von Programmzeilen auf dem Bildschirm in ATARI BASIC und Turbo BASIC XL 1.5.

Bei LIST in Turbo BASIC werden die Schleifen 2 Leerschritte eingerückt, was die Lesbarkeit des Programms erhöht. Ebenso kann dem Computer mitgeteilt werden, daß er von einer bestimmten Zeilennummer bis zum Programmende alle Zeilen zeigen soll. Zu diesem Zweck muß nach der Angabe der Anfangszeilennummer ein Komma folgen. In diesem Fall nimmt der Computer 32767 als Endzeilennummer (voreingestellter Wert) an.
LIST 3000,

Beispiel :LIST .... Zeigt das gesamte Programm auf dem Bildschirm.
LIST 10 .... Zeigt Zeile 10.
LIST 10,100 .... Zeigt alle alle Zeilen von 10 bis 100.
LIST "P:" .... Gibt das gesamte Programm auf dem Drucker aus.
LIST "P:",10,100 .... Gibt die Zeilen 10 bis 100 auf dem Drucker aus.
LIST 3000, ... Zeigt alle Zeilen von 3000 bis zum Ende an.
LIST "D:PROGRAMM.LST" .... Speichert das Programm im ATASCII-Format auf eine Diskette in Laufwerk 1. Es wird mit der ENTER-Instruktion wieder gelesen. Das so abgespeicherte Programm kann mit einem Textverarbeitungsprogramm bearbeitet werden
LIST "D2:PROGRAMM.LST",10,100 Speichert die Zeilen 10 bis 100 im ATASCII-Format auf eine Diskette in Laufwerk 1.
LIST "C:" .... Speichert das gesamte Programm im ATASCII-Format auf eine Kassette.

Siehe auch *L

LOAD
Systeminstruktion

Definition:Lädt ein Programm aus einem Peripheriegerät in dem Speicher.

Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :LOAD "gerät[n]:[dateispezifikation]"
oder statt LOAD die Abkürzung LO.

Anmerkung :gerät muß das Kürzel für ein Peripheriegerät wie Diskettenlaufwerk, Kassettenlaufwerk oder RAMDisk (D, C oder D8) sein.
n ist die Laufwerksnummer, von 1 bis maximal 8.

dateispezifikation ist ein beliebiger Dateiname, bestehend aus höchstens 8 Zeichen plus 3 Zeichen für den Anhang, der durch einen Punkt vom Dateinamen getrennt wird.

Beispiel :LOAD "C:"
LOAD "D:PROGRAMM.BAS"
LOAD "D8:TEST"

LOCATE
Grafikinstruktion

Definition:Findet Daten, die an einer bestimmten Stelle auf dem Bildschirm speichert sind.

Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :LOCATE x,y,d
oder statt LOAD die Abkürzung LOC.

Anmerkung :Verwendet in den Grafikbetriebsstufen 0 bis 2 Zeichen und in den Grafikbetriebsstufen 3 bis 11 Farbwerte.

Beispiel :LOCATE 100,100,D

Untersucht die Bildschirmposition 100,100. Der festgestellte Wert wird dann der Variablen D zugewiesen.
In den Stufen 0 bis 2 gibt die LOCATE-Anweisung den ATASCII-Wert an.
In den Stufen 3 bis 11 wird der COLOR-(Farb-)Wert an D übergeben.
Für x und y dürfen auch berechnete Werte benutzt werden.

LOCATE PEEK(86)*256+PEEK(85),PEEK(84),D

Die LOCATE-Instruktion wird auch benutzt, um eine Kollision im Zusammenhang mit Player/Missile-Grafiken zu erkennen.

LOCK und UNLOCK
I/O-Instruktion

Definition:Schützt eine Datei vor versehentlichem überschreiben, bzw. entfernt diesen Schutz wieder.

Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :LOCK "D[n]:*.*"
UNLOCK "D[n]:*.*"

Anmerkung :n ist die Laufwerksnummer, die nicht angegeben werden muß, falls Diskettenlaufwerk 1 angesprochen werden soll.
. repräsentiert den Dateinamen mit Anhang (in diesem Falle Platzhalter (Wildcards).
Falls eine Datei mit LOCK vor versehentlichem überschreiben geschützt wurde, kann man mit UNLOCK diesen Schutz wieder aufheben.

Beispiel :LOCK "D2:*.BAS"
Schützt alle Dateien mit dem Anhang BAS in Laufwerk 2 gegen überschreiben.

UNLOCK "D2:*.BAS"
Hebt den Schutz aller Dateien mit dem Anhang BAS in Laufwerk 2 wieder auf.

LOG
Arithmetische Funktion

Definition:Gibt den natürlichen Logarithmus einer Zahl (Basis e).
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :v=LOG(x)
Anmerkung :Diese Funktion ist reziprok zur EXP-Funktion.
x ist ein numerischer Wert oder ein berechneter Wert größer als Null.
Beispiel :

```
10 X=1.753: P=3
20 A=P*LOG((INT(X*10)+.5)/10)
30 PRINT A
```

LPRINT
I/O-Instruktion

Definition:Gibt etwas über den Drucker aus.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :LPRINT var[,var][,var] ...
oder statt LPRINT die Abkürzung LP.
Anmerkung :Bei der LPRINT-Instruktion ist es nicht notwendig, einen Datenkanal zum Drucker mit OPEN und CLOSE zu öffnen und zu schließen.
Beispiel :

```
LPRINT A$
LPRINT A;A$
LPRINT B,X$
LPRINT "Dies ist ein Test."
LP. "LP. oder LPRINT ist dasselbe."
```

--
Bearbeitungsinstruktion

Definition:Bestimmte REM-Funktion.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :-- (Zwei Minuszeichen)
Anmerkung :Genauso wie bei REM wird alles nach -- überschlagen und vom Programm nicht beachtet. Wenn das Programm mit LIST wieder angeschaut wird, erscheinen dreißig statt nur zwei Minuszeichen.
Diese Funktion verbraucht sogar weniger Speicherplatz als eine REM-Zeile ohne Text.
Diese Funktion gibt Ihnen die Möglichkeit, bestimmte zusammengehörige Programmteile, wie zum Beispiel Schleifen und Prozeduren, deutlich in einem Programm hervorzuheben.

MOD
Arithmetische Funktion

Definition:Gibt den Restwert nach einer Division.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :v=a MOD b
Anmerkung :a MOD b ist die verkürzte Schreibweise von:
a-b*TRUNC(a/b)
MOD ist die Abkürzung von Modulo.
Beispiel :PRINT 14 MOD 8 ergibt 6
PRINT 14/8 ergibt 1.75

PRINT 27 MOD 2 ergibt 1
PRINT 27/2 ergibt 13.5

Dies berechnet sich wie folgt:

14:8 =1.75 dies abgeschnitten ergibt 1
8*1 =8
14-8 =6
also : 14 MOD 8 ergibt 6.

27:2 =13.5 dies abgeschnitten ergibt 13
13*2 =26
27-26=1
also : 27 MOD 2 ergibt 1.

MOVE
Bearbeitungsinstruktion

Definition:Blocktransfer
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :[-]MOVE quelle,bestimmung,zähler
Anmerkung :quelle ist die erste zu lesende Speicheradresse, die kopiert werden soll.
bestimmung ist die Bestimmung, zu der der Inhalt der Quelle kopiert werden soll.
zähler ist die Anzahl der zu kopierenden Speicheradressen ab quelle.
Die Übersetzung ins ATARI BASIC kann wie folgt aussehen:

```
FOR I=0 TO ZAEHLER-1
  POKE BESTIMMUNG+I,PEEK(QUELLE+I)
NEXT I
```

Falls vor der MOVE-Instruktion ein Minuszeichen (-) steht, findet die Ausführung in umgekehrter Reihenfolge statt. Erst das letzte Byte, usw., usw...
Dies macht es möglich, Daten zu einer höheren Adresse zu verschieben, ohne, daß bei einer Überlappung (falls quelle+zähler > bestimmung) etwas gelöscht wird.
Die Instruktion:

-MOVE quelle,bestimmung,zähler

heißt dann in ATARI BASIC:

```
FOR I=COUNT-1 TO 0 STEP -1
  POKE BESTIMMUNG+I,Peek(quelle+i)
NEXT I
```

name
Spezielle Funktion

Definition:Benennt eine Zeile.
Version :ATARI BASIC Turbo BASIC Compiler
                      ***   ***

Syntax :# name
Anmerkung :# ist ein verpflichtendes Zeichen.
name ist ein beliebiger Stringausdruck.
Diese Funktion dient als Erkennungsmerkmal für die Sprungbefehle:

```
GO# name
ON var GO# name,name...
TRAP# name
RESTORE# name
```

Sehen Sie sich auch die jeweiligen Erklärungen dazu an.

NEW
Systeminstruktion

Definition:Löscht ein Programm aus dem Speicher und setzt alle Variablen auf Null.
Version :ATARI BASIC Turbo BASIC Compiler
         ***   ***   ***
Syntax :NEW
Anmerkung :Alle Variablen werden auf Null gesetzt. Gleichfalls wird die Tonausgabe gestoppt.
NEW kann sowohl im direkten Modus, als auch in einem Programm benutzt werden.
Beispiel :READY

```
READY
 NEW
READY
```

Das Programm, das im Speicher war, ist nun gelöscht worden

NEXT
Programminstruktion

Definition:Beendung einer FOR-NEXT-Schleife.
Version :ATARI BASIC Turbo BASIC Compiler
         ***   ***   ***
Syntax :FOR var=var TO var

```
FOR var=var TO var
  .
  .
  .
 NEXT var
```

Anmerkung :Siehe FOR.
NEXT darf auch mit N. abgekürzt werden.

NOTE
I/O-Instruktion

Definition:Bestimmt den Ort des nächsten zu lesenden oder zu schreibenden Bytes auf einer Diskette.

Version	:ATARI BASIC	Turbo BASIC	Compiler
	***	***	***

Syntax :NOTE #n,sektor,byte
oder statt NOTE die Abkürzung NO.

Anmerkung :# ist verpflichtend.
n gibt eine Kanalnummer von 1 bis 8 an.
sektor ist eine Variable, in die die Sektornummer eingesetzt wird.
byte ist eine Variable, in die die Bytenummer eingesetzt wird, die gelesen oder geschrieben werden soll.

ON und EXEC
Programminstruktion

Definition:Springt, abhängig vom Wert eines Ausdruckes, zu einer bestimmten Prozedur.

Version	:ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax :ON numaus EXEC prname, pname...

Anmerkung :numaus ist ein numerischer Ausdruck vom Typ Integer zwischen 0 und 255.
prname ist der Prozedurname, der jeweils angesprochen wird.
(numaus=1 => prname1; numaus=2 => prname2; numaus=3 => prname3...)

Ist der numerische Ausdruck unwahr (FALSE), dann wird das Programm mit der nächsten Zeile fortgeführt.
Nachdem die Prozedur, die mit prname aufgerufen wurde, abgearbeitet wurde, wird das Programm ebenfalls mit der nächsten Zeile fortgeführt, die nach ON ... EXEC folgt.

Beispiel :

```
10 CLS
20 REPEAT
30   GET TASTE
40 UNTIL TASTE>48 OR TASTE<52
50 ON TASTE-48 EXEC PROZEDUR_1, PROZEDUR_2, PROZEDUR_3
60 END
100 PROC PROZEDUR_1
 .
 .
 .
```

Zeilen 20 bis 40: warten auf einen Tastendruck. Dies solange wiederholen, bis der ATASCII-Wert der gedrückten Taste gleich 49, 50 oder 51 ist. Das entspricht den Tasten 1, 2 und 3.
Zeile 50: Falls die Taste 1 gedrückt wurde, wird zur Prozedur 1 gesprungen, falls die Taste 2 gedrückt wurde, wird zur Prozedur 2 gesprungen, und so weiter.

Siehe auch ON ... GOSUB.

ON und GO#
Programminstruktion

Definition:Springt, abhänging vom Wert eines Ausdruckes, zu einer Zeile mit einem bestimmten Namen (Label).
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :ON numaus GO# name, name ...
Anmerkung :numaus ist ein numerischer Ausdruck vom Typ Integer, größer Null und höchstens so groß wie die Summe der name-Ausdrücke nach GO#.
ist ein zwingend vorgeschriebenes Zeichen. Es zeigt an, daß keine Zeilennummer, sondern ein Labelname (String) folgt.
name ist ein Stringausdruck, der auf eine besonders benannte Zeile verweist.
Beispiel :

```
10 CLS
20 REPEAT
30   POSITION 2,2: PRINT "Wählen Sie! (1-3)";
40   GET TASTE
50 UNTIL TASTE>48 OR TASTE<52
60 ON TASTE-48 GO# WAHL1, WAHL2, WAHL3
70 END
100 #WAHL1
 .
 .
 .
200 #WAHL2
 .
 .
 .
300 #WAHL3
 .
 .
 .
```

Siehe auch ON ... GOSUB, ON ... GOTO und GO#.

ON und GOSUB sowie ON und GOTO
Programminstruktion

Definition:Springt, abhängig vom Wert eines Ausdruckes, zu einer bestimmten Zeilennummer.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :ON numaus GOTO zeile[,zeile] ...
ON numaus GOSUB zeile[,zeile] ...
Anmerkung :numaus ist ein numerischer Ausdruck vom Typ Integer zwischen 0 und 256.
zeile ist die Zeilennummer, zu der jeweils gesprungen werden soll.

Falls der Ausdruck numaus unwahr (FALSE) ist, wird das Programm mit der nächsten Zeile fortgesetzt. ON ... GOTO oder ON ... GOSUB ist die ideale Art, um über ein Auswahlmenü ein Unterprogramm anzuspringen und wieder zurückzukehren.

Beispiel :
```
100 OPEN #1,4,0,"K:"
110 GET #1,TASTE
120 IF TASTE<48 OR TASTE>52 THEN 100
130 ON TASTE-48 GOSUB 1000,2000,3000,4000
140 CLOSE #1
150 GOTO 100
1000 REM Beginn des ersten Unterprogramms
1001 REM Wenn TASTE-48=1
.
.
.
1999 RETURN
.
.
```

OPEN
I/O-Instruktion

Definition:Bereitet ein Peripheriegerät für die Ein- oder Ausgabe vor.

Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :OPEN #iocb,bearb,0,"gerät[n]:[dateispezifikation]"

Anmerkung :# ist ein verpflichtendes Zeichen.
iocb gibt die Kanalnummer (1 bis 7) an, die geöffnet werden soll.
bearb stellt den Bearbeitungscode dar. Verwendet werden folgende Codes:

4 INPUT nur lesen
6 DIRECTORY Inhaltsverzeichnis einer Diskette
8 OUTPUT nur schreiben
9 APPEND Zusatz zum Dateiende
12 UPDATE lesen und schreiben, Veränderung einer Datei

0 ist zwingend vorgeschrieben. Sie hat zwar keinen Einfluß auf die Operation, muß aber aus internen Gründen angeführt werden.

gerät ist das Peripheriegerät, das angesprochen (geöffnet) werden soll. Erlaubt sind:

K Tastatur .... (engl.:Keyboard)
P Drucker .... (engl.:Printer)
C Kassettenrekorder .... (engl.:Cassette Recorder)
D Diskettenlaufwerk .... (engl.:Diskette Drive)
E Bildschirmbearbeitungsgerät. (engl.:Editor)
R RS232 Serielle Schnittstelle (von RS232)
S Bildschirm.................. (engl.:Screen)

n ist die Kanalnummer, sie ist nicht unbedingt vorgeschrieben. Der voreingestellte Wert beträgt 1. Erlaubt sind Werte von 1 bis 8. Sie werden nur bei Verwendung von Diskettenlaufwerken (D) oder der RS232-Schnittstelle (R) benutzt. Falls R benutzt wird, muß ein serielles Schnittstellenprogramm geladen werden, um die serielle Schnittstelle zu öffnen.

dateispezifikation muß bei Verwendung eines Diskettenlaufwerkes angegeben werden. Sie besteht aus dem Dateinamen und dem Anhang des zu öffnenden Datenbestandes auf der Diskette.

: ist ein vorgeschriebenes Zeichen.

Beispiel :Eingabebearbeitung

```
 10 OPEN #1,4,0,"K:"           :REM Tastatur öffnen
 20 GET #1,TASTE               :REM Auf Tastatureingabe warten
 30 CLOSE #1                   :REM Tastatur schließen
 40 PRINT TASTE                :REM ATASCII-Wert anzeigen
```

Ausgabebearbeitung

```
 10 OPEN #1,8,0,"D:TEST.DAT"   :REM Datei öffnen
 20 FOR I=1 TO 100             :REM Schleife von 1 bis 100
 30   PRINT #1;I               :REM In die Datei schreiben
 40 NEXT I                     :REM 1 bis 100
 50 CLOSE #1                   :REM Datei schließen
```

Eingabebearbeitung in Turbo BASIC

```
 10 GET TASTE
 20 PRINT TASTE
```

Sie bewirkt dasselbe wie das allererste Beispiel in ATARI BASIC.

Eingabebearbeitung über das Diskettenlaufwerk

```
 10 OPEN #1,4,0,"D:TEST.DAT"
 20 FOR I=1 TO 100
 30   INPUT #1,X
 40   PRINT X;
 50 NEXT I
 60 CLOSE #1
```

Die Zahlen von 1 bis 100 des vorigen Beispiels werden von der Diskette zurückgeholt und hintereinander auf den Bildschirm geschrieben. Wenn die folgenden Zeilen hinzufügen, werden die Zahlen über den Drucker ausgegeben.

```
 15 OPEN #2,8,0,"P:"
 45   PRINT #2;X
 70 CLOSE #2
```

Das Inhaltsverzeichnis der Diskette

```
 10 OPEN #1,6,0,"D:*.*"        :REM Inhaltsverzeichnis öffnen
 20 TRAP 60                    :REM Falls Fehler gehe zu Zeile 60
 30 INPUT #1,FILE$             :REM Einen String von Diskette holen
 40 PRINT FILE$                :REM Den String anzeigen
 50 GOTO 30                    :REM Wieder einen String holen ...
 60 CLOSE #1                   :REM Inhaltsverzeichnis schließen
```

In Turbo BASIC kann das obenstehende Programm auch durch den Gebrauch des Befehls DIR ersetzt werden.

. in Zeile 10 sind Platzhalter, sogenannte Wildcards. Diese können ersetzt werden, wenn zum Beispiel Dateien mit gleichen Namenseigenschaften ausgeben will.
*.BAS meint alle Dateien mit dem Anhang BAS (BASIC-Programme).
TEST.* meint alle Dateien, die zu TEST gehören.
Und so weiter.

PADDLE
Steuergeräte-Funktion

Definition:Gibt den Wert eines Drehreglers (Paddle).

Version	:ATARI BASIC	Turbo BASIC	Compiler
	***	***	***

Syntax :x=PADDLE(n)

Anmerkung :x ist eine numerische Variable.
n ist die Drehreglernummer. Angesprochen werden können Drehregler von 0 bis 3. Die Werte, die x ergibt liegen zwischen 1 und 228. Eine Drehung nach links (also gegen den Uhrzeigersinn) ergibt steigende Werte.

Beispiel :IF PADDLE(0)>14 THEN 100
Ein Drehregler wird nur von wenigen Spielprogrammen abgefragt. Der Joystick ist weiter verbreitet, obwohl es viele interessante Anwendungen für Drehregler gibt.

PAINT
Grafikinstruktion

Definition:Füllt eine geschlossene Figur mit der durch FCOLOR gewählten Farbe.

Version	:ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax :PAINT x,y

Anmerkung :x und y sind die Koordinaten auf einem Grafikbildschirm (Spalte und Zeile) innerhalb der zu füllenden Figur.
Diese Instruktion kann Figuren praktisch jeder Form einfärben. Angemerkt sei, daß PAINT sehr viel Speicherplatz benötigt. Falls dieser zu klein wird, folgt die Fehlermeldung ERROR-2 MEM.
Die PAINT-Instruktion macht Gebrauch von einigen sehr schnellen Zeichenfunktion, da hierbei viele nebeneinander liegende Punkte verhältnismäßig einfach berechnet werden können.

Beispiel :
```
10 GRAPHICS 24
20 SETCOLOR 2,4,0: COLOR 3
30 CIRCLE 160,80,40
40 PAINT 160,80
50 GOTO 50
```

Sehen Sie sich bitte auch das Beispiel unter FILLTO an.

PAUSE
Programminstruktion

Definition:Unterbricht die Programmausführung für n 50stel Sekunden.

Version	:ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax :PAUSE n

Anmerkung :n ist die Zeiteinheit, die ein Programm unterbrochen werden soll, bevor es weitergeführt wird.
PAUSE braucht weniger Speicherplatz und ist genauer als eine leere FOR-NEXT-Schleife.

Beispiel :PAUSE 100
Zwei Sekunden Unterbrechung.

PEEK
Spezielle Funktion

Definition:Gibt den Inhalt einer bestimmten Speicheradresse.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :a=PEEK(n)
Anmerkung :n ist ein Wert vom Typ Integer zwischen 0 und 65535. n bezeichnet die zu lesende Speicheradresse. Der Wert, der a zugewiesen wird, ist ebenfalls ein integrer Wert, er liegt zwischen 0 und 255. PEEK ist das Gegenteil von POKE (siehe dort). PEEK hat keinen Einfluß auf die zu lesende Speicheradresse, im Gegensatz zu POKE.
Beispiel :Mit dem folgenden kleinen Programm kann der Inhalt jeder Speicheradresse betrachtet werden.

```
10 FOR N=1 TO 65535
20   PRINT N;" ";PEEK(N)
30 NEXT N
```

Die ermittelten Werte können auch in Berechnungen benutzt werden.

```
A=PEEK(560)+256*PEEK(561)
```

Siehe auch DPEEK.

PLOT
Grafikinstruktion

Definition:Setzt einen Punkt oder ein Zeichen an eine bestimmte Stelle auf dem Bildschirm.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :PLOT x,y
oder als Abkürzung für PLOT nur PL.
Anmerkung :x und y repräsentieren die Koordinaten der zu PLOTtenden Bildschirmposition. Diese sind immer abhängig von der Grafikbetriebsstufe, die gewählt wird.
Beispiel :Mit dem folgenden Beispiel werden im Grafikmodus 5 willkürlich Punkte auf den Bildschirm gesetzt.

```
5 GRAPHICS 5+16           :REM Grafikmodus 5 ohne Textfenster
10 FOR I=1 TO 25          :REM 25 Punkte
20   X=INT(RND(0)*80)+1   :REM Zufällige Werte für
30   Y=INT(RND(0)*48)+1   :REM X und Y
40   PLOT X,Y             :REM Einen Punkt auf den Schirm setzen
50 NEXT I                 :REM Schluß der Schleife
60 GOTO 60                :REM Auf die BREAK-Taste warten
```

Die Zeilen 20 und 30 können in Turbo BASIC XL 1.5 verkürzt werden.

```
20 X=RAND(80)
30 Y=RAND(48)
```

POINT
I/O-Instruktion

Definition:Teilt dem Disketten-Betriebssystem (DOS) mit, wo sich das nächste zu lesende oder zu schreibende Byte befindet.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :POINT #n,sektor,byte
oder statt POINT die Abkürzung P.
Anmerkung :n ist eine Kanalnummer von 1 bis 7. Der Kanal muß zuvor geöffnet werden.
sektor ist die betreffende Sektornummer und
byte ist die betreffende Bytenummer auf der Diskette.

POKE
Bearbeitungsinstruktion

Definition:Schreibt einen Wert in eine Speicheradresse.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :POKE n,m
Anmerkung :n gibt die Speicheradresse an, in die der Wert geschrieben werden soll und muß ein integrer Wert zwischen 0 und 65535 sein.
m ist die Information, die in die bestimmte Speicherstelle geschrieben werden soll. Der Wert muß zwischen 0 und 255 liegen und vom Typ Integer sein.
Das Gegenstück zu POKE ist die Funktion PEEK.

Achtung!
BASIC kontrolliert die Speicheradressen nicht. POKEn Sie also nicht auf gut Glück in den BASIC Stapelspeicher, die Variablentabelle oder in Ihr Programm!
Beispiel :POKE 82,4
Setzt den Wert 4 in die Speicherstelle 82. Dies hat zur Folge, daß die linke Schreibbegrenzung auf 4 eingestellt wird.

Sehen Sie sich bitte für eine ausführliche Erklärung der Speicheradressen den Anhang A an.

Siehe auch DPOKE

POP
Programminstruktion

Definition:Verläßt ein mit GOSUB aufgerufenes Unterprogramm vorzeitig und macht den letzten GOSUB-Befehl ungültig, beziehungsweise verläßt eine FOR-NEXT-Schleife vorzeitig und macht die letzte ausgeführte FOR-Anweisung ungültig.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :POP
Anmerkung :POP zeigt Wirkung in Unterprogrammen, die mit GOSUB oder EXEC (Turbo BASIC) aufgerufen wurden, oder in Schleifen mit:
FOR .... NEXT
REPEAT ... UNTIL (Turbo BASIC)
WHILE ... WEND (Turbo BASIC)

Falls es notwendig ist, die oben genannten Schleifen oder Unterprogramm-Konstruktionen vorzeitig zu verlassen, sollte dies mit der POP-Instruktion geschehen.

Anmerkung :
```
 5 DIM A$(100)
 10 FOR I=1 TO 100
 20   INPUT A$
 30   IF A$="STOP" THEN POP: GOTO 60
 40   PRINT A$
 50 NEXT I
 60 IF A$="STOP" THEN PRINT "Sie haben STOP eingetippt!"
 70 END
```
Das obenstehende Beispiel fordert hundertmal die Eingabe einer Stringvariablen. Falls STOP eingegeben wird, wird die FOR-NEXT-Schleife beendet und zur Zeile 60 gesprungen.

Siehe auch EXIT.

POSITION
Grafikinstruktion

Definition:Positioniert die Schreibmarke auf eine bestimmte Bildschirmposition.

Version	:ATARI BASIC	Turbo BASIC	Compiler
	***	***	***

Syntax :POSITION x,y
oder statt POSITION die Abkürzung POS.

Anmerkung :x ist die betreffende Spalte.
y ist die betreffende Zeile.

Beispiel :
```
 10 FOR X=1 TO 20
 20   POSITION X,X:PRINT "TEST"
 30 NEXT X
```
Diese Beispiel schreibt das Wort TEST auf den Bildschirm, beginnend in der ersten Spalte und Zeile. Danach in der zweiten Spalte und Zeile. Und so weiter. Das Ergebnis ist ein schräg nach unten laufendes Bild mit dem Wort TEST.

POSITION ist auch in anderen Grafikbetriebsstufen zu verwenden.
```
 10 GRAPHICS 18
 20 FOR X=3 TO 8
 30   POSITION X,X: PRINT #6;"TEST"
 40 NEXT X
 50 GOTO 50
```

Siehe auch XIO.

PRINT
I/O-Instruktion

Definition:Speichert Daten auf den Bildschirm oder ein anderes Gerät.

Version	:ATARI BASIC	Turbo BASIC	Compiler
	***	***	***

Syntax :PRINT [#n;]["folge von ausdrücken"][;] ...
oder statt PRINT die Abkürzungen PR. oder ?

Anmerkung :n ist eine Kanalnummer (IOCB) von 1 bis 7, falls die PRINT-Instruktion sich auf eine Datei oder eine Grafikstufe bezieht.
und ; sind dann vorgeschrieben.

folge von ausdrücken darf jedes Zeichen sein, das mit der Tastatur eingegeben werden kann. Ebenso darf jede numerische oder Stringvariable mit PRINT gespeichert werden.

Beispiel :PRINT bildet eine leere Zeile ab.

```
PRINT          bildet eine leere Zeile ab.
PRINT A        bildet den Wert der Variable A ab
PRINT A$       bildet den Inhalt der Variable A$ ab
PRINT "TEST!"  bildet TEST! ab
```

Wenn PRINT sich auf ein Peripheriegerät bezieht, sehen Sie sich bitte das Beispiel bei OPEN an.

Wenn PRINT sich auf eine Grafikstufe bezieht, sehen Sie sich bitte das Beispiel 2 bei POSITION an.

Es können mehrere Ausdrücke oder/und Variablen nach einer PRINT-instruktion bearbeitet werden, getrennt durch ein Komma oder ein Semikolon.
Ein Komma nach einem Ausdruck oder einer Variablen speichert den darauffolgenden Ausdruck oder die darauffolgende Variable im Abstand der jeweiligen Tabulatoreinstellung. Ein Semikolon plaziert die Ausdrücke oder Variablen ohne Zwischenraum hintereinander.

```
READY
PRINT "TEST","TEST"
TEST    TEST
READY
PRINT "TEST";"TEST"
TESTTEST
READY
```

Nach der PRINT-Instruktion können auch Berechnungen ausgeführt werden.

```
READY
 10 X=5
 20 PRINT X+5,X-5,X*(-5)
RUN
 10      0      -25
READY
```

Angemerkt sei noch, daß PRINT eine der wichtigsten Instruktionen in BASIC ist. Mit dieser Instruktion wird die Möglichkeit geschaffen, etwas auf dem Bildschirm mitzuteilen, oder diese Information an ein externes Gerät zu senden.

PROC
Programminstruktion

Definition:Zeigt den Beginn eines Unterprogramms (Prozedur) an.
Version :ATARI BASIC   Turbo BASIC ***   Compiler ***
Syntax :PROC name
Anmerkung :name ist der Prozedurname, mit der die Prozedur aufgerufen werden kann.
Nähere Erklärungen finden Sie unter EXEC und ON...EXEC.

PTRIG
Steuergeräte-Funktion

Definition:Gibt den Wert des Feuerknopfes von einem Drehregler an.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :a=PTRIG(n)
Anmerkung :n ist ein Wert von 0 bis 3 und gibt den betreffenden Drehregler an.
a hat den Wert 0, wenn der Feuerknopf gedrückt wird und 1, wenn er nicht gedrückt wird.

0 : Feuerknopf betätigt 1 : Feuerknopf im Ruhezustand

PUT
I/O-Instruktion

Definition:Sendet ein Byte zu einem bestimmten Peripheriegerät.
Version :ATARI BASIC Turbo BASIC Compiler
*** 1 *** 1,2 ***
Syntax :1. PUT #n,byte
2. PUT n
Anmerkung :zu 1.
ist ein zwingend vorgeschriebenes Zeichen
n gibt eine Kanalnummer von 1 bis 7 an, die zu einem mit OPEN geöffneten Gerät oder einer so geöffneten Datei verweist.

zu 2.
n ist ein Wert vom Typ Integer zwischen 0 und 255.
Mit dieser Instruktion kann ein Zeichen auf dem Bildschirm angezeigt werden. Da hier im Gegensatz zu 1. der IOCB-Anzeiger #n fehlt, wird automatisch IOCB #0 geöffnet, also der Bildschirm beziehungsweise das Bildschirmbearbeitungsgerät.

PUT n entspricht PRINT CHR$(n).

Beispiel :zu 1.
```
10 OPEN #2,8,0,"D:DATEI.DAT"
20 FOR BYTE=1 TO 100
30   PUT #2,BYTE
40 NEXT BYTE
```
Mit diesem Beispiel werden 100 Zeichen in die Datei DATEI.DAT geschrieben.
Um die Informationen aus DATEI.DAT wieder zurückzuholen, sehen Sie sich bitte das Beispiel unter GET an.

PUT #2,BYTE entspricht PRINT #2,A

zu 2.
```
10 FOR ZEICHEN=0 TO 255
20   PUT ZEICHEN
30 NEXT ZEICHEN
```
Mit diesem Beispiel werden alle Zeichen des ATARI Rechners auf dem Bildschirm dargestellt.

Siehe auch BPUT und %PUT.

%PUT
I/O-Instruktion

Definition:Sendet ein Byte zu einem bestimmten Peripheriegerät.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :%PUT #n,byte
Anmerkung :% zeigt dem Interpreter an, daß das zu sendende Byte in komprimierter Form (6-Byte-Format) gespeichert werden soll. Dies bewirkt eine schnellere Datenübertragung und spart Platz auf dem Speichermedium.
ist ein verpflichtendes Zeichen.
n ist eine Kanalnummer von 1 bis 7 und verweist zu einem mit OPEN geöffneten Kanal (siehe auch OPEN).
byte ist die Zahl, die gespeichert werden soll.
Beispiel :

```
10 OPEN #1,8,0,"D:DATEI.DAT"
20 FOR I=1 TO 1000
30   BYTE=RAND(100)
40   %PUT #1,BYTE
50 NEXT I
60 CLOSE #21
```

Dieses Beispiel schreibt 1000 zufällige Zahlen zwischen 0 und 100 in die geöffnete Datei DATEI.DAT.

Siehe auch %GET.

RAD
Trigonometrische Funktion

Definition:Alle trigonometrischen Funktionen,die nach RAD folgen, werden im Bogenmaß verarbeitet.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :RAD
Anmerkung :RAD wandelt alle trigonometrischen Funktionen ins Bogenmaß. Dies geschieht solange, bis die DEG-Instruktion auszuführen ist. Dann folgt die Verarbeitung im Winkelmaß.

RAND
Arithmetische Funktion

Definition:Erzeugt eine ganzzahlige Zufallszahl zwischen einschließlich 0 und ausschließlich n.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :v=RAND(n)
Anmerkung :n ist ein numerischer Ausdruck, der die Obergrenze der zu erzeugenden Zufallszahl darstellt.
RAND(n) ist die verkürzte Schreibweise für TRUNC(RND(0)*n).
Beispiel :

```
PRINT RAND(100)
oder
V=RAND(100): PRINT V
```

Siehe auch RND, FRAC und TRUNC.

READ
Bearbeitungsinstruktion

Definition:Liest einzeln die Information in einer DATA-Zeile und weist sie einer Variablen zu.

Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :READ var[,var] ...

Anmerkung :var ist eine numerische oder Stringvariable, der die Information aus einer DATA-Zeile zugewiesen wird.
Die READ-Anweisung arbeitet mit einem Zeiger (Pointer), so daß der Interpreter bei der nächsten READ-Anweisung weiß, wo er fortzufahren hat. Deshalb müssen nicht alle Daten einer DATA-Zeile mit einem mal eingelesen werden.
Der Zeiger kann jederzeit neu definiert werden, so daß er wieder auf die erste oder eine andere Zeile zeigt. Sehen Sie sich dazu bitte die RESTORE-Instruktion an.

Beispiel :
```
 10 DIM NAME$(20),ORT$(20)
 20 PRINT "Name","Ort","Telefonnummer"
 30 READ NAME$,ORT$,TELEFON
 40 PRINT NAME$,ORT$,TELEFON
 50 END
 60 DATA ABBUC,HERTEN,39623
RUN
 NAME    ORT     TELEFON
 ABBUC   HERTEN  39623
READY
```

REM
Bearbeitungsinstruktion

Definition:Erlaubt es, Anmerkungen oder dergleichen in das Programm einzufügen.

Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :REM Anmerkung
oder statt REM die Abkürzungen R. oder [leerzeichen].

Anmerkung :Jedes Zeichen nach REM wird von BASIC nicht beachtet. Der häufige Einsatz von REM macht ein Programm gerade in ATARI BASIC für einen selbst oder andere durchschaubarer, jedoch wird es dann etwas langsamer und verbraucht mehr Speicherplatz.
REM darf direkt hinter einer Zeilennummer stehen, aber auch nach einer Funktion, Anweisung oder dergleichen, wenn es durch einen Doppelpunkt abgetrennt wird.

Beispiel :
```
10 REM Hier beginnt das Programm
20 REM Nun folgt eine Schleife
30 FOR ...
40   GOSUB 500: REM Sprung zu einem Unterprogramm
 .
 .
100 NEXT ...
110 END: REM Dies ist das Ende
500 REM Dies ist das Unterprogramm
 .
 .
 .
599 RETURN :REM Sprung zurück ins Hauptprogramm
```

RENAME
I/O-Instruktion

Definition:Gibt einer Datei auf einer Diskette einen anderen Namen.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :RENAME "D[n]:alt,neu"
Anmerkung :n gibt die Laufwerksnummer an. Sie ist bei Verwendung von Laufwerk 1 nicht vorgeschrieben.
alt ist der alte Dateiname mit Anhang.
neu ist der neue Dateiname mit Anhang.
Beispiel :RENAME "D:MEINPROG.TUR,DEINPROG.TUR"

Dies ist vergleichbar mit

XIO 32,#7,0,0,"D[n]:alt,neu"

und der RENAME-Option im DOS-Menu.

RENUM
Systeminstruktion

Definition:Numeriert Programmzeilen um.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :RENUM alt,neu,erh
Anmerkung :alt ist die alte Zeilennummer, ab der die Umnumerierung beginnen soll.
neu ist der neue Anfangswert für die Zeilennummer; praktisch die Nummer, die die Stelle der alten einnimmt.
erh ist die Schrittgröße (Erhöhung), mit der die Umnumerierung stattfinden soll.

Die Zeilennummern vor alt bleiben unverändert. Zeilennummern nach Instruktionen wie GOTO, GOSUB, TRAP, RESTORE, LIST, DEL, ON...GOTO und ON...GOSUB werden mit umnumeriert. Bei berechneten Sprüngen versagt RENUM; so zum Beispiel bei GOTO A, GOSUB 100+10*A, TRAP A+B, RESTORE A*10+1000 und so weiter.
Falls nach einer solchen Sprunginstruktion eine Zahl folgt, wird sie von RENUM mit umnumeriert, der nachfolgende Programmteil bis zu einem Doppelpunkt oder dem Zeilenende jedoch bleibt unverändert; GOTO 100+10*A wird also wie GOTO 100 behandelt.
Derartige berechnete Sprungbefehle sind jedoch in einer strukturierten Programmiersprache wie Turbo BASIC überflüssig.
Wenn nach dem Sprungbefehl ein (Prozedur-)Name oder ein Ausdruck in Klammern folgt, wird er überhaupt nicht verändert.
Beispiel :Das folgende Beispiel numeriert ein Programm ab Zeile 130 um. Der neue Anfangswert statt 130 wird 200 und die Schrittgröße beträgt 20. Die Zeilen ab 200 (die alte 130) werden also stets um 20 erhöht; 200, 220, 240 und so weiter.
RENUM 130,200,20

oder das gesamte Programm (falls die erste Zeilennmummer 10 ist):
RENUM 10,10,10

Alle drei Werte müssen angegeben werden.

REPEAT und UNTIL
Programminstruktion

Definition:Wiederholt Anweisungen, bis eine Bedingung erfüllt ist.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :REPEAT anw UNTIL ausdr
Anmerkung :anw darf jede beliebige Instruktion, Berechnung oder jeder Ausdruck sein.
ausdr ist ein Ausdruck, der wahr (TRUE) sein muß, um die Schleife zu verlassen.
Die Anweisungen innerhalb der Schleife werden mindestens einmal ausgeführt, da die Abfrage, ob der Ausdruck wahr ist, erst am Ende der Schleife geschieht.
Beispiel :

```
10 CLS
20 REPEAT
30   PRINT "Drücken Sie eine Taste!"
40   GET TASTE: PRINT TASTE
50 UNTIL TASTE=65
60 PRINT "Sie haben die richtige Taste gefunden!"
70 PRINT "Sie ist :";STR$(TASTE)
```

Siehe auch WHILE...WEND.

RESTORE
Bearbeitungsinstruktion

Definition:Verweist zu einer bestimmten Zeile mit der Anweisung DATA, deren Informationen mit Hilfe von READ gelesen werden können.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :RESTORE [zeile]
oder statt RESTORE die Abkürzung RES.
Anmerkung :zeile gibt die betreffende Zeilennummer an und ist nicht verpflichtend. Falls RESTORE ohne die Angabe einer Zeilennummer benutzt wird, wird der Zeiger für den READ-Befehl stets auf die erste Information in der ersten Zeile mit einer DATA-Anweisung gesetzt.
Bei RESTORE 150 zum Beispiel wird bei der nächsten Leseanweisung READ mit der ersten Information in Zeile 150 begonnen; auch wenn sich vor Zeile 150 noch andere DATA-Zeilen befinden, die noch nicht gelesen wurden.
Beispiel :

```
READY
 10 READ A
 20 RESTORE 150
 30 READ B
 40 PRINT A,B
 100 DATA 150,200,250
 110 DATA
  .
  .
  .
 150 DATA 123,456,789
RUN
 150     123
READY
```

RESTORE #...
Bearbeitungsinstruktion

Definition:Verweist zu einer benannten Zeile mit der Anweisung DATA, deren Informationen mit Hilfe von READ gelesen werden können.

```
Version   :ATARI BASIC   Turbo BASIC   Compiler
                            ***           ***
```

Syntax :RESTORE # name

Anmerkung :name ist der Name der Zeile (Label), auf die RESTORE verweist.

Sehen Sie sich dazu bitte die Anmerkungen unter GO #..., TRAP #... und RESTORE an.

RETURN
Programminstruktion

Definition:Beendet ein Unterprogramm, daß mit GOSUB aufgerufen wurde, und springt zur nächsten Anweisung, die auf GOSUB folgt.

```
Version   :ATARI BASIC   Turbo BASIC   Compiler
               ***          ***           ***
```

Syntax :RETURN
oder statt RETURN die Abkürzung RET.

Anmerkung :Dies ist die einzige Möglichkeit, ein Unterprogramm nach vollständiger Ausführung aller ihrer Anweisungen zu beenden.
Eine andere, nicht so elegante Art, ein Unterprogramm zu verlassen, ist die POP-Instruktion (siehe dort). Sie ist jedoch unstrukturiert und sollte nur in Notfällen benutzt werden.

Beispiel :

```
5 DIM A$(FRE(0)-1000)
10 INPUT A$
20 IF A$<>"STOP" GOSUB 100: GOTO 10
30 PRINT A$
40 END
100 REM Anfang des Unterprogramms
110 A=A+LEN(A$)
120 PRINT "Gesamtheit der eingegebenen Zeichen: ";A
130 RETURN
```

Siehe auch EXEC, PROC und ENDPROC.

RND
Arithmetische Funktion

Definition:Gibt eine zufällige Zahl zwischen 0 und 1.

```
Version   :ATARI BASIC   Turbo BASIC   Compiler
               *** 1        *** 1,2       ***
```

Syntax :1.
A=RND(0)

2.
A=RND[(n)]

Anmerkung :zu 1.
Dies ist die vorgeschriebene Syntax in ATARI BASIC.

zu 2.
In Turbo BASIC ist die Angabe zwischen den eckigen Klammern nicht verpflichtend. Um Speicherplatz zu sparen, genügt RND.

Der Wert, der der Variablen A zugewiesen wird, liegt zwischen 0 und ausschließlich 1.
Wenn eine Zufallszahl größer als 1 erzeugt werden soll, muß das Ergebnis mit dem höchsten gewünschten Wert multipiziert werden.
Wenn ganze Zahlen (also vom Typ Integer) erzeugt werden sollen, muß das Ergebnis mit INT verarbeitet werden.

Beispiel :zu 1.

```
A=RND(0)             :REM Eine Zahl zwischen 0 und 1
A=RND(0)*8           :REM Eine Zahl zwischen 0 und 8
A=INT(RND(0)*8)+1 :REM Eine Zahl zwischen 0 und 9 vom Typ Integer
```

zu 2.

```
A=RND
A=RND*8
A=INT(RND*8)+1
```

RUN
Systeminstruktion

Definition:Führt ein BASIC-Programm aus.

Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :RUN ["gerät[n]:[dateispezifikation]"]
oder statt RUN die Abkürzung RU.

Anmerkung :gerät gibt das Gerät an, von der das zu startende Programm geholt werden soll. D und C sind die Kennungen für das Disketten- und Kassettenlaufwerk.
n gibt die Laufwerksnummer für die Diskette an und muß zwischen 1 und 8 liegen. Wenn sie weggelassen wird, wird Laufwerk 1 angesprochen.
: ist ein verpflichtendes Zeichen, wenn eine Laufwerkskennung angegeben wird.
" ebenso.
dateispezifikation stellt den Programmnamen mit seinem Anhang dar. Er wird bei Verwendung eines (RAM-)Diskettenlaufwerkes gebraucht.

RUN setzt alle Variablen auf null und löscht die Dimensionierung von Variablenfeldern und Stringvariablen.

Beispiel :

```
10 FOR X=1 TO 10
20   PRINT "X=";X
30 NEXT X
RUN
X=1
X=2
 .
 .
```

und so weiter.

RUN "C:"
Lädt das nächste Programm von einer Kassette und führt es aus.

RUN "D:MEINPROG.BAS"
Lädt das Programm MEINPROG.BAS von einer Diskette in Laufwerk 1 und führt es aus.

SAVE
Systeminstruktion

Definition:Sendet ein BASIC-Programm zu einem Peripheriegerät, das es abspeichert.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :SAVE "gerät[n]:[dateispezifikation]"
oder statt SAVE die Abkürzung S.
Anmerkung :gerät gibt das Gerät an, zu der das Programm gesandt werden soll. D und C sind die Kennungen für das Disketten- und Kassettenlaufwerk.
n gibt die Laufwerksnummer für die Diskette an und muß zwischen 1 und 8 liegen. Wird sie weggelassen, so wird Laufwerk 1 angesprochen.
dateispezifikation stellt den Programmnamen und Anhang dar, mit dem das Programm im (RAM-)Diskettenlaufwerk abgespeichert werden soll.
Bei Verwendung eines Kassettenlaufwerkes ist es nicht notwendig, eine Dateispezifikation anzugeben, da diese nicht wie bei einer Diskette in einem Inhaltsverzeichnis abgelegt wird.
Beim Laden von einer Kassette muß das Band so positioniert werden, daß der Anfang des vorher abgespeicherten Programms genau am Schreib/Lesekopf liegt.
Bei Verwendung eines Diskettenlaufwerkes ist die Angabe eines Programmnamens notwendig, da dieser zusammen mit einem Zeiger, wo das Programm auf der Diskette beginnt, im Inhaltsverzeichnis abgelegt wird.
Beispiel :SAVE "C:"
SAVE "D2:MEINPROG.BAS"

SETCOLOR
Grafikinstruktion

Definition:Bestimmt den Farbton und die Helligkeit des gewählten Farbregisters
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :SETCOLOR register,farbton,helligkeit
oder statt SETCOLOR die Abkürzung SE.
Anmerkung :register ist das Farbregister Werte von 0 bis 4 sind erlaubt.
farbton ist ein Wert von 0 bis 15 für den Farbton (=16 Farben).
helligkeit gibt die Helligkeit der Farbe mit Werten von 0 bis 15 an (=16 Helligkeiten).
Eine kleine Rechnung zeigt, daß der ATARI Computer 256 (16*16) Farben darstellen kann.

Die SETCOLOR-Tabelle mit voreingestellten Werten:

Register	Farbe	Helligkeit	Name
0	2	8	Orange
1	12	10	Grün
2	9	4	Dunkelblau
3	4	6	Rosarot
4	0	0	Schwarz

Die Farbwahlmöglichkeiten mit den entsprechenden Nummern sind:

0 Grau	1 Gold	2 Orange	3 Orange-Rot
4 Rose	5 Violett-Rot	6 Violett-Blau	7 Blau-Rot
8 Blau	9 Hellblau	10 Türkis	11 Blau-Grün
12 Grün	13 Gelb-Grün	14 Orange-Grün	15 Hellorange

SGN
Arithmetische Funktion

Definition:Gibt das Vorzeichen eines numerischen Ausdruckes.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :var=SGN(x)
Anmerkung :var ist eine numerische Variable.
x darf jeder beliebige numerische Ausdruck sein.
Falls x positiv ist, wird der Variablen der Wert (+)1 zugewiesen.
Falls x negativ ist, wird der Variablen der Wert -1 zugewiesen.
Falls x gleich Null ist, wird der Variablen der Wert 0 zugewiesen.
Beispiel :ON SGN(X)+2 GOSUB 1000,2000,3000
Wenn x im negativen Bereich liegt, wird zum Unterprogramm ab Zeile 1000 gesprungen. Wenn x jedoch gleich Null ist, wird das Unterprogramm ab Zeile 2000 abgearbeitet. Bei einem positiven Wert von x wird schließlich zum Unterprogramm ab Zeile 3000 verzweigt.

SIN
Trigonometrische Funktion

Definition:Gibt den Sinus eines Wertes.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :var=SIN(x)
Anmerkung :var ist eine numerische Variable.
x ist ein numerischer Ausdruck im Bogenmaß, dessen Sinus berechnet werden soll.
Möchte man vom Winkel- in das Bogenmaß umrechnen, so multipliziert man mit PI/180 (PI=3.14593).
Beispiel :

```
READY
 10 PI=3.14593
 20 WINKEL=90
 30 BOGEN=WINKEL*PI/180
 40 PRINT SIN(BOGEN)
RUN
 1
READY
```

SOUND
Toninstruktion

Definition:Gibt einen Ton aus dem Lautsprecher des Fernsehgerätes oder des Monitiors aus.

Version :ATARI BASIC Turbo BASIC Compiler
*** 1 *** 1,2 ***

Syntax :1.
SOUND kanal,frequenz,verzerrung,lautstärke
oder statt SOUND die Abkürzung SO.

2.
SOUND

Anmerkung :zu 1.
Aktiviert einen der vier Tonkanäle und läßt dadurch über den Lautsprecher des Monitiors oder TV-Gerätes ein akustisches Signal erklingen. Das Signal wird solange ausgesandt, bis ein neuer SOUND-Befehl für diesen Tonkanal, NEW, RUN oder END ausgeführt wird.
In ATARI BASIC muß nach dem SOUND-Befehl die Angabe von vier Werten geschehen. Diese Werte dürfen Ziffern, sowie numerische Variablen oder Ausdrücke sein.

kanal gibt einen der Tonkanäle von 0 bis 3 an.
frequenz gibt die Frequenz an. Die Auflösung beträgt 8 Bit (0 bis 255), daraus folgt eine berechnete Frequenz (in Hertz) von 63921/(2*frequenz+2). Dieser Wert stammt aus einer amerikanischen Veröffentlichung und unterscheidet sich eventuell von der europäischen Norm. Die Ursache hierfür ist der Unterschied zwischen der amerikanischen NTSC- und der europäischen PAL-Norm. Die Veröffentlichung von ATARI gibt 31960/frequenz+1 als Frequenz an.
verzerrung bestimmt den Wert der Verzerrung des betreffenden Tones. Der hier einzusetzende Wert muß zwischen 0 und 14 liegen, wobei nur gerade Zahlen "saubere" Töne erzeugen. Bei Angabe ungerader Zahlen treten andere Verzerrungen auf.
lautstärke gibt die Lautstärke eines Tones an und darf zwischen 0 und 15 liegen. Je höher der Wert, desto lauter der Ton. Die Angabe von 0 stellt die Tonausgabe auf diesem Tonkanal ein.
Der Lautstärkewert aller Tonkanäle zusammen sollte 32 nicht übersteigen, da sonst der Lautsprecher anfangen könnte, zu brummen.

Sehen Sie sich dazu bitte auch die Tabelle der Tonwerte mit den dazugehörigen Tönen auf einer Klaviatur an.

zu 2.
SOUND ohne die Angabe der unter 1. beschriebenen Parameter bewirkt die Einstellung der Tonausgabe auf allen 4 Tonkanälen. Es ist die verkürzte Schreibweise von FOR I=0 TO 3: SOUND I,0,0,0: NEXT I.

Beispiel :

```
10 FOR I=1 TO 10
20   READ Frequenz
30   SOUND 1,Frequenz,14,12
40   FOR WARTE=1 TO 90: NEXT WARTE
50   SOUND 1,0,0,0
60   FOR WARTE=1 TO 90: NEXT WARTE
70 NEXT I
80 DATA 255,243,230,217,204,193,182,173
```

Angemerkt sei außerdem, daß die ATARI Home Computer 3,5 Oktaven darstellen können.
Siehe auch DSOUND.

SQR
Arithmetische Funktion

Definition:Gibt die positive Quadratwurzel eines positiven Wertes.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :A=SQR(x)
Anmerkung :x muß größer oder gleich Null sein.
Beispiel :

```
READY
 10 X=10
 20 PRINT X,SQR(X)
RUN
 10      3.162278
READY
```

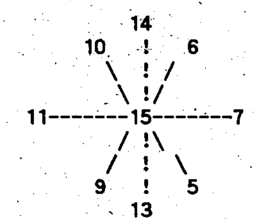
STATUS
I/O-Instruktion

Definition:Gibt den Statuswert eines durch IOCB geöffneten Peripheriegerätes und weist diesen einer numerischen Variable zu.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :STATUS #n,var
oder statt STATUS die Abkürzung ST.
Anmerkung :Der Statuswert wird von einem näher zu spezifizierenden Gerät gelesen.
n gibt die IOCB-Kanalnummer an und darf einen Wert von 1 bis 7 haben.
var ist die Variable, der der Statuswert zugewiesen wird.

STICK
Steuergeräte-Funktion

Definition:Gibt einen Wert, der die Stellung des Steuerhebels darstellt.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :x=STICK(n)
Anmerkung :n gibt den betreffenden Steuerhebel (Joystick) an. 0 steht für Joystickport 1 (die erste Buchse an der rechten Gehäuseseite), 1 steht für Joystickport 2 (die folgende Buchse).

Nun folgen die Werte, die diese Funktion erzeugen kann und die zugehörigen Stellungen des Steuerhebels:



15 : Steuerhebel im Ruhezustand
14 : Steuerhebel in Richtung Norden / nach vorn
13 : Steuerhebel in Richtung Süden / nach hinten
11 : Steuerhebel in Richtung Westen / nach links
7 : Steuerhebel in Richtung Osten / nach rechts
6 : Steuerhebel in Richtung Nordosten / nach rechtsvorne
5 : Steuerhebel in Richtung Südosten / nach rechtshinten
9 : Steuerhebel in Richtung Südwesten / nach linkshinten
10 : Steuerhebel in Richtung Nordwesten / nach linksvorne

Beispiel :

```
10 GRAPHICS 5+16
20 X=40: Y=24
30 X=STICK(0)
40 IF X<>15 THEN GOSUB 100
50 PLOT X,Y
60 GOTO 10
100 IF X=14 THEN Y=Y-1
110 IF X=10 THEN Y=Y-1: X=X-1
120 IF X=6 THEN Y=Y-1: X=X+1
130 IF X=13 THEN Y=Y+1
140 IF X=9 THEN Y=Y+1: X=X-1
150 IF X=5 THEN Y=Y+1: X=X+1
160 IF X=11 THEN X=X-1
170 IF X=7 THEN X=X+1
180 RETURN
```

STOP
Programminstruktion

Definition:Unterbricht ein Programm und kehrt in den direkten Modus zurück.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :STOP
Anmerkung :Falls STOP innerhalb eines Programmes steht, wird die Ausführung an dieser Stelle unterbrochen. Eventuell geöffnete Datenkanäle werden nicht geschlossen (siehe END, NEW und RUN). Ebenso wird die Tonausgabe nicht abgestellt.
Das Programm kann mit Eingabe der CONT-Instruktion aus dem direkten Modus fortgesetzt werden.
Beispiel :IF A>100 THEN STOP
Falls A größer 100 ist, wird das Programm unterbrochen und eine Meldung mit Angabe der Zeilennummer erscheint auf dem Bildschirm.

STR$
Stringfunktion

Definition:Wandelt eine Zahl in einen String um.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :X$=STR$(x)
Anmerkung :x darf jeder beliebige numerische Ausdruck sein.
Der numerische Ausdruck wird nach einer Umwandlung mit STR$ nicht mehr als Zahl oder numerischer Ausdruck erkannt.
Das Gegenstück zu STR$ ist die Funktion VAL (siehe VAL).
Beispiel :X$=STR$(100)
X$ wird nun "100" zugewiesen, so daß gilt:
X$="100"

```
10 GRAPHICS 0
20 PRINT "Geben Sie eine Zahl ein! ";: INPUT A
30 PRINT "Die Zahl ";A;" als String ";STR$(A)
```

STRIG
Steuergeräte-Funktion

Definition:Gibt einen Wert, der den Zustand des Feuerknopfes an einem Steuerhebel darstellt.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :a=STRIG(n)
Anmerkung :n stellt die Nummer des Steuerhebels dar, dessen Feuerknopf abgefragt werden soll (0 oder 1; siehe STICK).
a ist eine Variable, der eine 0 zugewiesen wird, wenn der betreffende Feuerknopf gedrückt wird. Im Ruhezustand beträgt dieser Wert 1.

0 : Feuerknopf gedrückt 1 : Feuerknopf im Ruhezustand
Beispiel :Zwei Programme, die die Randfarbe des Bildschirms durch einen Druck auf den Feuerknopf verändern. Beispiel 1 läuft unter ATARI BASIC und Beispiel 2 unter Turbo BASIC XL 1.5.

1.
```
10 GRAPHICS 0
20 POKE 752,1
30 A=STRIG(0)
40 IF A=0 THEN GOSUB 100
50 POKE 712,FARBE
60 GOTO 20
100 FARBE=INT(RND(0)*254)+1
110 RETURN
```

Erläuterungen:
Zeile 10 : Den Bildschirm löschen
Zeile 20 : Die Darstellung der Schreibmarke abschalten (0=an)
Zeile 30 : Den Wert des Feuerknopfes an Joystick 1 feststellen
Zeile 40 : Das Unterprogramm ab Zeile 100 anspringen, wenn der Feuerknopf gedrückt wurde
Zeile 50 : Einen Farbwert in die Speicheradresse des Bildschirmrahmens setzen
Zeile 60 : Zurück zu Zeile 20 springen

Zeile 100 : Eine zufällige Zahl zwischen 0 und 255 für die Farbe erzeugen
Zeile 110 : Zurück zum Hauptprogramm springen

2.
```
10 CLS
20 POKE 752,1
30 DO
40   A=STRIG(0)
50   IF A=0
60     EXEC FARBE_SETZEN
70     POKE 712,FARBE
80   ENDIF
90 LOOP
100 PROC FARBE_SETZEN
110 FARBE=INT(RND*254)+1
120 ENDPROC
```

Beispiel 1 arbeitet auch unter Turbo BASIC, aber Beispiel 2 nicht unter ATARI BASIC. Die Zeilen 50, 60 und 70 können auch durch
```
50 IF A=0 THEN EXEC FARBE_SETZEN
60 POKE 712,FARBE
```
ersetzt werden.

Es fällt auf, das Beispiel 2 deutlicher und lesbarer als Beispiel 1 ist.

TEXT
Grafikinstruktion

Definition:Stellt Text in einem Grafikbildschirm dar.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :TEXT x,y,ausdr
Anmerkung :x gibt die x-Koordinate auf einem Grafikbildschirm an (Spalte). y gibt die y-Koordinate auf einem Grafikbildschirm an (Zeile). ausdr stellt einen Stringausdruck dar, der zwischen Anführungszeichen stehen muß. Ein numerischer Ausdruck ist auch erlaubt und wird ohne Anführungszeichen angegeben.

x und y stellen immer die Position des ersten auszugebenden Zeichens (linksoben) an. Die Position wird in Bildpunkten angegeben. Die TEXT-Instruktion macht von einigen sehr schnellen Algorithmen Gebrauch, die viele nebeneinander liegende Bildpunkte verhältnismäßig einfach berechnen.

Beispiel :
```
10 GRAPHICS 8
20 TEXT 50,90,"Turbo BASIC"
30 TEXT 70,95,1000
40 GOTO 40
```

Siehe auch das Beispiel unter FILLTO.

TIME
Variable

Definition:Enthält die Zeit der internen Uhr (RTCLOCK).
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :v=TIME
Anmerkung :v ist die Variable, der die Zeit der internen Uhr zugewiesen wird. Die Zeit wird in 1/50 Sekunden registriert. Der Wert wird aus den Adressen 18, 19 und 20 gelesen.
Beispiel :PRINT TIME gibt die Zeit aus den Adressen 18 bis 20

TIME$
Spezielle Funktion

Definition:Enthält die Zeit in Form eines Strings.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

Syntax :1.
V$=TIME$

2.
TIME$=V$

Anmerkung :zu 1.
Enthält die Zeit in Form eines 6 Stellen zählenden Strings. Die Stringform sieht wie folgt aus:
HHMMSS
HH : Stunden von 00 bis 23
MM : Minuten von 00 bis 59
SS : Sekunden von 00 bis 59

zu 2.
Diese Form dient dazu, eine Zeit an TIME$ zu übergeben (die Uhr zu stellen). Die Variable TIME läßt sich nicht so leicht verändern; man müßte die Zeit mühsam in die Adressen 18 bis 20 einsetzen (POKEn). Mit TIME$ hingegen kann man dies sehr leicht bewerkstelligen.
Die Uhr, die durch TIME$ eingestellt wird, läuft nicht immer ohne Abweichungen. Das liegt daran, daß die Bildschirmfrequenz nicht exakt 50 Hertz beträgt. TIME$ wird von der Bildschirmfrequenz abgeleitet.

Beispiel :
```
 5 DIM V$(6),A$(2)
 10 INPUT "Stunden  00...23 ";A$
 20 EXEC ZEIT
 30 INPUT "Minuten  00...59 ";A$
 40 EXEC ZEIT
 50 INPUT "Sekunden 00...59 ";A$
 60 EXEC ZEIT
 70 DO
 80   POSITION 15,12: PRINT "Die Zeit ist: ";TIME$
 100 LOOP
 110 PROC ZEIT
 120   V$(LEN(V$)+1)=A$
 130 ENDPROC
```

```
10 CLS: DIM V$(6)
20 INPUT "Geben Sie die Zeit ein (HHMMSS)! ";V$
30 TIME$=V$: POKE 752,1
40 DO
50   POSITION 15,12: PRINT "Die Zeit ist: ";TIME$
60 LOOP
```

TIME$ kann nicht dimensioniert werden; TIME$ ist von vornherein dimensioniert.

TRACE
Systeminstruktion

Definition:Zeigt die Nummer jeder ausgeführten Zeile in eckigen Klammern auf dem Bildschirm.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :TRACE [zeichen]
Anmerkung :zeichen kann + oder - sein und ist nicht vorgeschrieben.

TRACE oder TRACE+
Schaltet den TRACE-Modus ein. Das bedeutet, daß die Nummer der Programmzeilen, die momentan abgearbeitet werden, in eckigen Klammern auf dem Bildschirm angezeigt werden.

TRACE-
Schaltet den TRACE-Modus aus. Außerdem wird der TRACE-Modus aufgehoben, wenn ein Fehler auftritt. Die Zeilennummern einer Prozedur oder einer Marke (Label) werden bei Aufruf mit EXEC oder GO# nicht ausgegeben.

TRAP
Programminstruktion

Definition:Fängt eine Fehlermeldung ab und führt das Programm mit einer bestimmten Zeile fort.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :TRAP zeile
oder statt TRAP die Abkürzung T.
Anmerkung :zeile gibt die Zeilennummer an, mit der das Programm fortgesetzt werden soll, wenn ein Fehler auftritt. Nachdem ein Fehler festgestellt wurde, wird die TRAP-Anweisung wieder ungültig. Wenn ein eventuell folgender Fehler auch mit TRAP bearbeitet werden soll, muß TRAP wieder spezifiziert werden.
TRAP kann im Programm auch ohne eine Fehlermeldung wieder abgestellt werden. Zu diesem Zweck muß für zeile ein Wert höher als 32767 angegeben werden. 40000 wird hierfür oft benutzt.
(TRAP 40000)
Beispiel :

```
10 TRAP 100
20 INPUT A
30 PRINT A
40 GOTO 20
100 PRINT "Es wurde keine Zahl eingegeben!"
110 GOTO 10
```

Wenn bei der Abfrage in Zeile 20 ein String eingegeben wird (zum

Beispiel HALT), stellt der Interpreter einen Fehler fest, der durch TRAP abgefangen wird. TRAP setzt das Programm dann mit einer vom Anwender/von der Anwenderin definierten Fehlermeldung in Zeile 100 fort.

PEEK(195) gibt die Fehlernummer an.
PEEK(187)*256+PEEK(186) gibt die fehlerhafte Zeile an.

Siehe auch TRAP#.

TRAP#...
Programminstruktion

Definition: Fängt eine Fehlermeldung ab und führt das Programm mit einer benannten Zeile fort.

Version	ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax : TRAP # name

Anmerkung : # ist ein vorgeschriebenes Zeichen. Es zeigt dem Interpreter an, daß ein Labelname statt einer Zeilennummer folgt.
name ist ein Name (String) für eine vorher benannte Zeile.

Siehe auch GO#... und RESTORE#... .

Beispiel :

```
10 TRAP # FEHLER
20 REPEAT
30   INPUT "Geben Sie eine Zahl ein: ";B
40   A=A+B
50 UNTIL A>100
60 PRINT A
70 END
100 #FEHLER
110 PRINT "Sie haben eine fehlerhafte Eingabe gemacht!"
120 GOTO 10
```

TRUNC
Arithmetische Funktion

Definition: Ermittelt den integren Wert vor dem Komma eines numerischen Ausdruckes.

Version	ATARI BASIC	Turbo BASIC	Compiler
		***	***

Syntax : v=TRUNC(ausdr)

Anmerkung : ausdr stellt einen numerischen Ausdruck dar.

TRUNC gibt den ganzzahligen Anteil einer Zahl. Dies ist die komplementäre Funktion zu FRAC.

Beispiel :

```
PRINT TRUNC(-0.3)  ergibt 0
PRINT TRUNC(-1.7)  ergibt -1
```

UINSTR
Stringfunktion

Definition:Sucht einen String ein einem längeren String ohne Rücksichtnahme auf Groß- und Kleinschreibung.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***
Syntax :A=UINSTR(A$,B$[,i])
Anmerkung :Siehe Anmerkungen zu INSTR.

Was diese Funktion von INSTR unterscheidet, ist, daß sie nicht auf Groß- oder Kleinbuchstaben achtet. So wird zum Beispiel bei der Suche nach MODEM auch Modem oder MoDeM gefunden. Ebenso darf der Text auch invers sein.
Beispiel :Siehe Beispiel bei INSTR.
Verändern sie in diesem Beispiel B$="vErs".
Als Ergebnis erhalten sich immer noch 18.

USR
Spezielle Funktion

Definition:Ruft von BASIC aus ein Unterprogramm in Maschinensprache auf.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***
Syntax :a=USR(adresse,argument[,argument...])
Anmerkung :adresse gibt die Speicheradresse an, ab der das Unterprogramm beginnt.
argument stellt eines oder mehrere Argumente dar, die in Form eines einzelnen Bytes an den Stapelspeicher übergeben werden.
Die Übergabe geschieht in umgekehrter Reihenfolge; also das letzte Argument zuerst, und so weiter.

Beim Aufruf des Unterprogramms wird von einer "Dummy"-Variable gebrauch gemacht. Sie hat auf die Ausführung der Funktion keinen Einfluß, es muß aber aus internen Gründen diese Syntax eingehalten werden (beim obigen Beispiel ist es die Variable a).

Erst nachdem alle Argumente vom Stapelspeicher geholt wurden, geht die Kontrolle wieder an BASIC zurück. Falls keine Argumente angegeben werden, wird eine Null an den Stapelspeicher übergeben.
Beispiel :A=USR(1536)
Die Argumente in Adresse 1536 werden ausgeführt; dies ist die sogenannte PAGE 6, eine leere Seite in der Speicherkarte. Hier können Maschinensprache-Unterprogramme abgelegt werden und zusammen mit BASIC (vollkommen unabhänging voneinander) ausgeführt werden.
So kann zum Beispiel eine kleine Hintergrundmelodie abgespielt werden, während das BASIC-Programm einige Berechnungen macht.

VAL
Stringfunktion

Definition:Wandelt einen String in eine Zahl um.
Version :ATARI BASIC Turbo BASIC Compiler
*** *** ***

Syntax :x=VAL(stringausdruck)
Anmerkung :Der Stringausdruck muß mit einer Zahl beginnen, sonst wird der Fehler 18 erzeugt.
Beispiel :READY

```
 10 DIM A$(10)
 20 A$="1000"
 30 PRINT VAL(A$)
RUN
 1000
READY

 10 DIM A$(10)
 20 A$="1HERTEN"
 30 PRINT VAL(A$)
RUN
 1
READY
```

VAL ist das Gegenstück von STR$ (siehe dort).

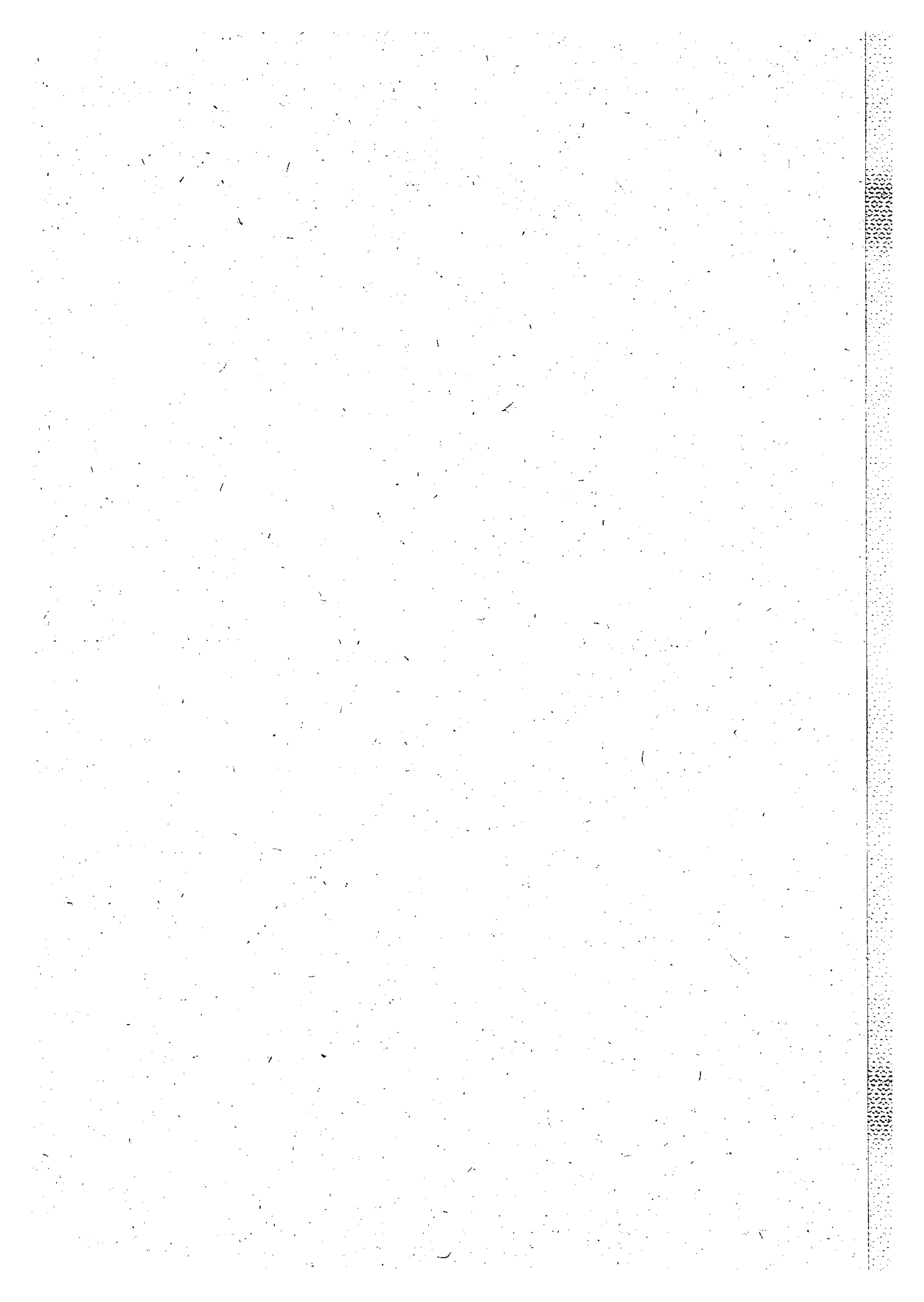
WHILE und WEND
Programminstruktion

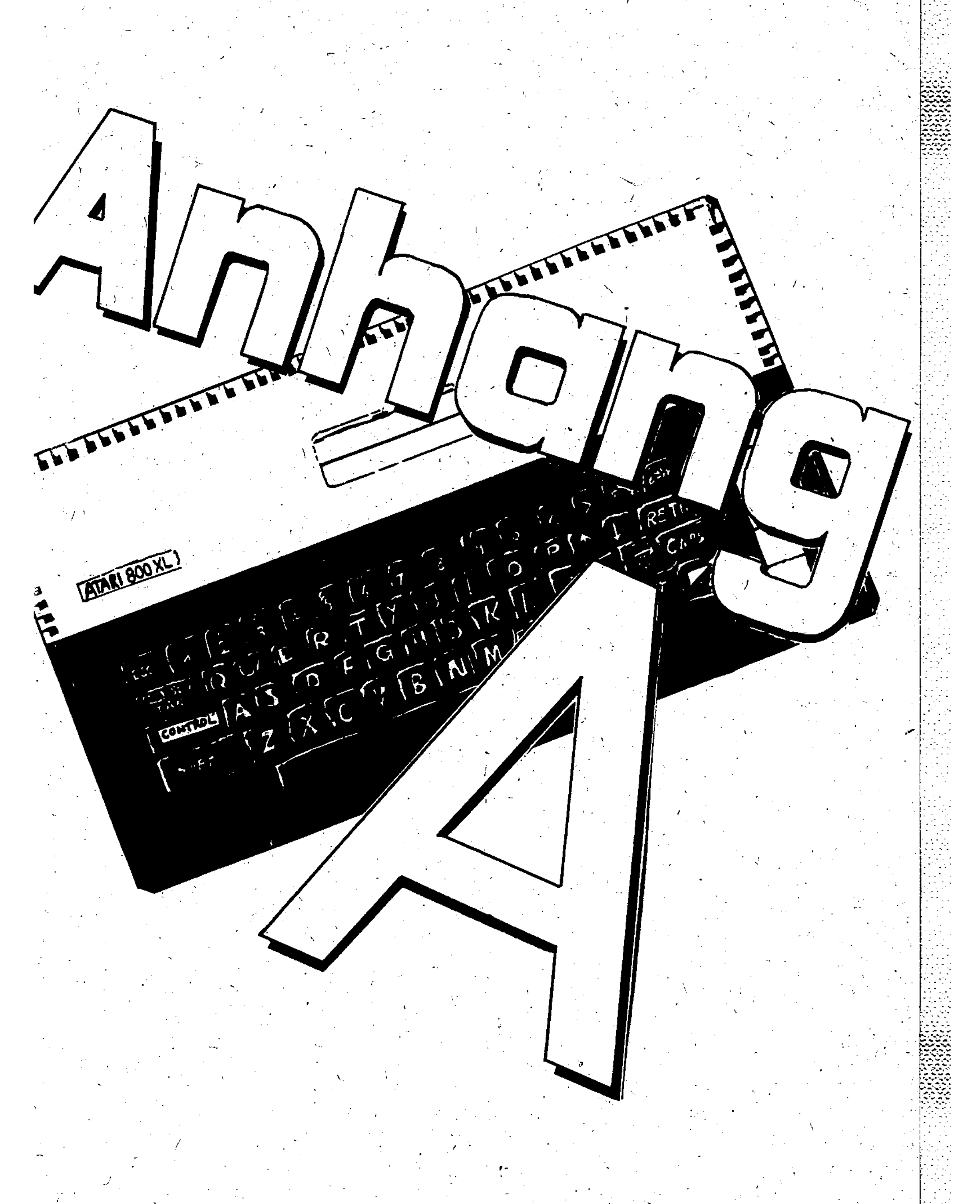
Definition:Wiederholt Anweisungen, solange bestimmte Bedingung erfüllt ist.
Version :ATARI BASIC Turbo BASIC Compiler
*** ***

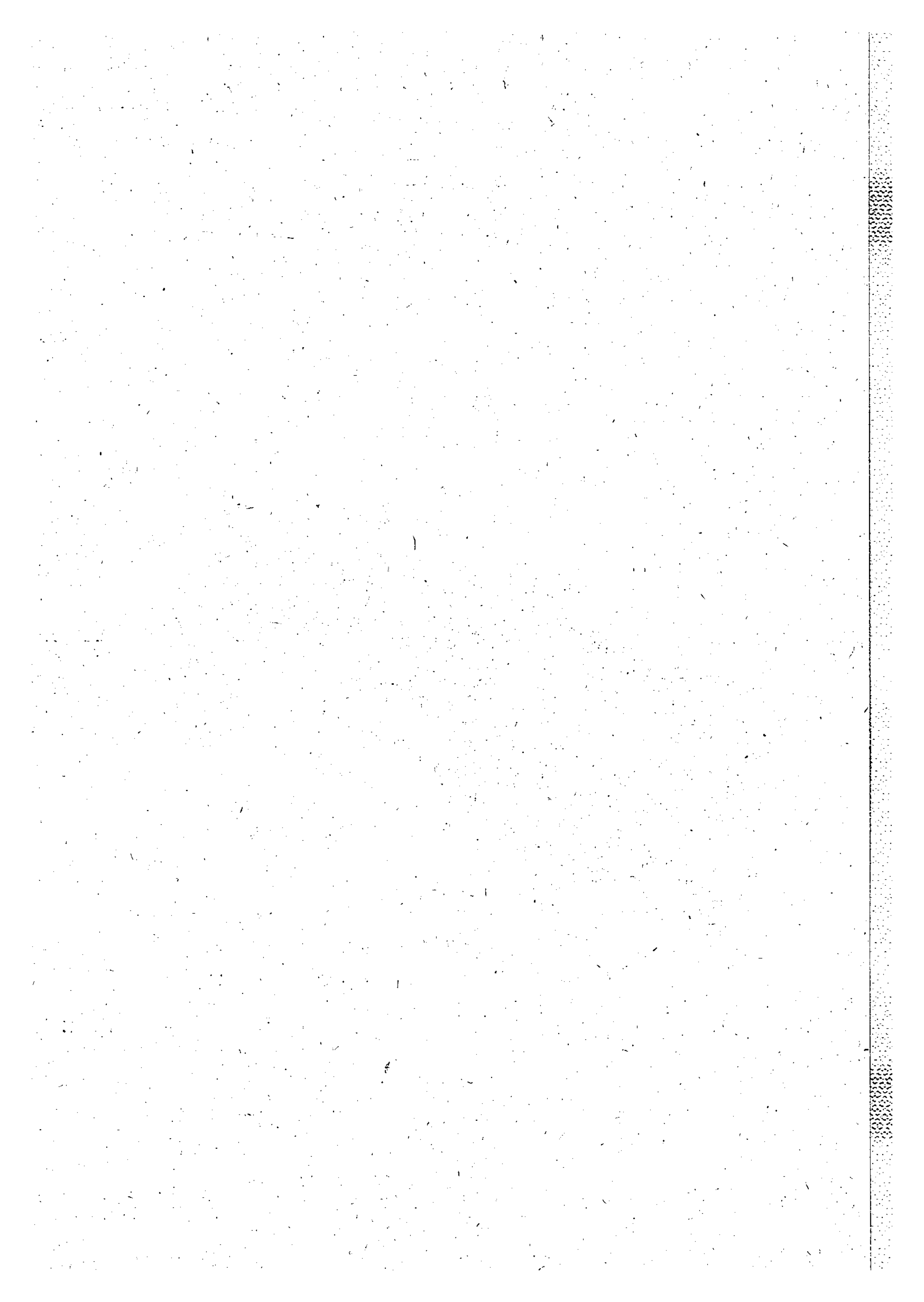
Syntax :WHILE ausdr ... anweisungen ... WEND
Anmerkung :ausdr ist ein Ausdruck, der wahr (true) sein muß, wenn die Schleife abgearbeitet werden soll.
Die Probe, ob die Bedingung erfüllt ist oder nicht, wird bei dieser Schleifenkonstruktion gleich zu Beginn gemacht. Das heißt, daß die Schleife kein einziges mal ausgeführt wird, wenn die Bedingung nicht erfüllt wird, sie also unwahr (FALSE) ist. In diesem Falle wird mit der Instruktion nach WEND fortgefahren.
anweisungen stellen Instruktionen dar, die ausgeführt werden, bis die Bedingung unwahr wird.
Beispiel :

```
 10 CLS
 20 EXEC FRAGE
 30 WHILE TASTE<>65
 40   PRINT TASTE
 50   EXEC FRAGE
 60 WEND
 70 PRINT "Sie haben die richtige Taste gefunden!"
 80 PRINT "Es war: ";STR$(TASTE)
 90 END
 100 PROC FRAGE
 110   PRINT "Betätigen Sie eine Taste!"
 120   GET TASTE
 130 ENDPROC
```

Vergleichen Sie dieses Programm bitte mit dem Beispiel unter REPEAT...UNTIL.




Anhang
A
ATARI 800 XL
CONTROL



ANHANG A: FEHLERMELDUNGEN

Erläuterungen zu der Frage, wann Fehler in BASIC auftreten, finden Sie im Hauptteil 2-5 unter dem Stichwort Interpreter.

Es ist möglich, eine Fehlermeldung, die das Programm normalerweise unterbricht, mit Hilfe des TRAP-Befehls zu unterdrücken (siehe Hauptteil 3). Andere BASIC-Funktionen, die sich mit Fehlern beschäftigen, sind ERR und ERL (siehe Hauptteil 3).

Die untenstehende Aufführung ist in ihrer Gesamtheit aus einem Artikel im ATARI Magazin der [ehemaligen] Stichting ATARI Gebruikers übernommen worden. Eine vollständigere Liste ist mir bis heute nicht untergekommen, und an dieser Stelle bedanke ich mich bei Wim Denie dafür, daß er sie mir zur Verfügung gestellt hat (ich habe einige Ergänzungen vorgenommen).

-	Definition	Fehlernummer
INSUFFICIENT MEMORY	SPEICHERPLATZ REICHT NICHT AUS	2
Es ist zuwenig Speicherplatz vorhanden, um noch neue Anweisungen, Stringvariablen oder Variablenfelder in diesem Umfang verarbeiten zu können. Halten Sie in Ihrem Programm nach überflüssigen Variablen Ausschau und löschen sie diese, oder bauen Sie Ihren Speicher aus.		
VALUE ERROR	WERTEBEREICH FALSCH	3
Sie haben eine zu große Zahl (im positiven oder negativen Bereich) benutzt. Es ist möglich, daß Sie eine Zeilennummer über 32767 angegeben haben, oder daß Sie mit einer PLOT- oder DRAWTO-Anweisung über die möglichen Bildschirm-Koordinaten hinausgegangen sind; siehe auch Fehlernummer 144.		
TOO MANY VARIABLES	ZU VIELE VARIABLEN	4
Das Programm soll in ATARI BASIC mehr als 128 und in Turbo BASIC mehr als 256 Variablen enthalten. Auch früher benutzte (zugewiesene) Variablen zählen hier mit. Legen Sie Ihr Programm mit LIST auf der Diskette ab, geben Sie NEW ein und holen Sie das Programm mit ENTER wieder in den Speicher. Die Variablentabelle ist nun gesäubert und enthält nur noch die in Programm vorkommenden Variablen.		
STRING LENGTH ERROR	STRINGLÄNGE FALSCH	5
Es wurde ein String dimensioniert und versucht, von einer Stelle zu lesen oder zu einer Stelle zu schreiben, die außerhalb der String-Dimension liegt. Dimensionieren Sie den String größer.		
OUT OF DATA	ZUWENIG DATEN	6
Es sind zu wenig Informationen in einer Zeile mit der Anweisung DATA enthalten, das heißt, daß mit READ mehr Daten gelesen werden sollen, als vorhanden sind. Verwirrend mag sein, daß die Fehlermeldung die Zeile mit der betreffenden READ-Anweisung bemängelt. Geben Sie soviele Daten an, daß jede Variable einen Wert erhält. Eine häufige Fehlerquelle ist die Eingabe eines Punktes in DATA-Zeilen anstelle eines Kommas. Die READ-Anweisung interpretiert dann 2 ganzzahlige Werte als eine reelle Zahl; das hat zur Folge, daß die Reihe der Daten an Ende um 1 Element zu kurz ist. Dieser Fehler stellt		

sich auch immer dann ein, wenn Sie sich mit der Schreibmarke über dem Wort READY befinden und die RETURN-Taste drücken. Der Computer legt dieses Wort als READ Y aus und teilt entrüstet mit, daß er nicht genug Informationen hat, um Y zu füllen.

NUMBER GREATER THAN 32767 — ZAHL GRÖSSER ALS 32767 — 7

Es wurde versucht, eine größere Zeilennummer als 32767 zu verwenden. Schauen Sie Ihre GOTO- und GOSUB-Anweisungen durch.

INPUT_STATEMENT ERROR — FALSCHER INPUTBEFEHL — 8

Dieser Fehler tritt oft auf, wenn versucht wird, einer numerischen Variablen einen Stringwert zuzuweisen, oder wenn die Eingabe einer Zahl erwartet wird und man nur die RETURN-Taste drückt (im Prinzip dasselbe).

ARRAY OR STRING DIM ERROR — DIMENSIONIERUNGSFEHLER — 9

Es wurde versucht, ein Variablenfeld mit mehr als 5461 Einträgen beziehungsweise einen String mit mehr als 32767 Zeichen Länge zu dimensioniern. Es ist auch möglich, daß Sie versucht haben, einen existenten String neu zu dimensionieren, beziehungsweise auf einen nicht dimensionierten String zu verweisen. Vergleichen Sie Ihre Variablen mit den dimensionierten Werten.

STACK OVERFLOW — STAPELSPEICHER ÜBERLAUF — 10

Es sind zuviele oder zu verschachtelte Unterprogramme im Stapelspeicher untergebracht.

FLOATING POINT OVERFLOW/UNDERFLOW — GLEITKOMMA ÜBERLAUF/UNTERLAUF — 11

Entweder wurde eine Division durch 0 versucht, oder das Ergebnis einer Berechnung ist zu groß beziehungsweise zu klein, um es darstellen zu können. Der Zahlenbereich der ATARI Computer reicht von -10 ^ 98 bis 10 ^ 98. Die kleinste darstellbare Zahl ist 1/10 ^ 98.

LINE NOT FOUND — ZEILE NICHT GEFUNDEN — 12

Es wurde zu einer nicht vorhandenen Zeilennummer verwiesen, zum Beispiel durch GOTO, GOSUB, nach einem THEN und so weiter.

NO MATCHING FOR — KEINE PASSENDE FOR-ANWEISUNG — 13

Dieser Fehler wird auch NEXT WITHOUT FOR (NEXT- OHNE FOR-ANWEISUNG) genannt. Während des Programmlaufes stieß der Interpreter auf eine NEXT-Anweisung, ohne vorher die benötigte FOR-Anweisung erhalten zu haben. Möglicherweise sind Sie beim Programmieren verschachtelter Schleifen durcheinander gekommen, oder Sie haben eine falsche Schleifenvariable gewählt. Dieser Fehler tritt auch auf, wenn innerhalb einer FOR...NEXT-Schleife der POP-Befehl unsachgemäß benutzt wird.

LINE TOO LONG — ZEILE ZU LANG — 14

Sie haben eine Programmzeile mit mehr als 114 Zeichen Länge oder mehr als 3 Bildschirmzeilen eingegeben. Wenn Sie die Anweisungen abkürzen, gestattet der Rechner mehr Zeichen. Wenn eine solche Zeile wieder aufgeLISTet wird, dürfen Sie in ihr nicht mehr RETURN betätigen.

GOSUB OR FOR LINE DELETED — GOSUB ODER FOR GELÖSCHT — 15

Die zu NEXT oder RETURN gehörenden Anweisungen FOR oder GOSUB wurden gelöscht.

RETURN ERROR — RETURN-FEHLER Eine RETURN ohne passendes GOSUB wurde gefunden. In einigen Fällen fehlt eine END-Anweisung nach dem Hauptprogramm.	16
SYNTAX ERROR — SYNTAX-FEHLER Sie haben eine für den Computer unverständliche Zeile eingegeben und sich an der Fehlermeldung nicht gestört. Es kann aber auch vorkommen, daß der Grund ein Hardwarefehler oder ein unbedachter POKE-Befehl ist.	17
INVALID STRING CHARACTER — STRINGBEGINN FALSCH Die Stringvariable in einer VAL-Funktion ist stellt keine Ziffer dar.	18
LOAD PROGRAM TOO LONG — LOAD-PROGRAMM ZU LANG Es steht zu wenig Speicher zur Verfügung, um ein Programm von Diskette oder Kassette vollständig zu laden.	19
DEVICE NUMBER ERROR — KANALNUMMER FALSCH Die benutzte Kanalnummer liegt nicht zwischen 1 und 7.	20
LOAD FILE ERROR — LADEBEFEHL FALSCH Sie haben versucht, ein Programm mit LOAD zu laden, daß nicht mit mit SAVE abgespeichert wurde, sondern mit CSAVE oder LIST. Laden Sie das Programm in diesem Fall mit CLOAD oder ENTER. Dieser Fehler tritt auch auf, wenn versucht wird, ein Maschinenspracheprogramm mit LOAD zu laden.	21
NEST ERROR — VERSCHACHTELUNGS-FEHLER Dieser Verschachtelungsfehler tritt auf, wenn das zu einem WHILE gehörende WEND nicht gefunden wird, oder das ENDIF zu einem IF, oder auch, nach *F+, das NEXT zu einem FOR. Dieser Fehler tritt auch auf, wenn eine FOR...NEXT-Schleife oder ein Unterprogramm unerlaubt beendet wird.	22
WEND WITHOUT WHILE — WEND OHNE WHILE In einer WHILE...WEND-Schleife fehlt die Anweisung WHILE. Dieser Fehler kann auftreten, wenn man mit einem GOTO-Befehl in eine solche Schleife springt. Sobald der Interpreter das WEND findet, erfolgt die Fehlermeldung.	23
UNTIL WITHOUT REPEAT — UNTIL OHNE REPEAT Dieser Fehler entspricht der Nummer 23, aber er kommt in REPEAT... UNTIL-Schleifen zum Tragen.	24
LOOP WITHOUT DO — LOOP OHNE DO Dieser Fehler entspricht der Nummer 23, aber er kommt in DO...LOOP-Schleifen zum Tragen.	25
EXIT ERROR — EXIT-FEHLER Es wurde versucht, mittels EXIT eine Schleife zu beenden, die garnicht existiert.	26
EXECUTING PROC ERROR — FEHLER BEIM AUSFÜHREN VON PROC Dieser Fehler wird dann gemeldet, wenn eine Prozedur ausgeführt werden soll, ohne daß sie mit EXEC aufgerufen wurde. PROC selber kann nämlich nicht ausgeführt werden; damit wird nur die Prozedur benannt. Diesen Fehler trifft man häufig im Zusammenhang mit unzu-	27

lässigen Sprungbefehlen an.

ENDPROC WITHOUT EXEC — ENDPROC OHNE EXEC Dieser Fehler entspricht sinngemäß der Nummer 23, tritt aber bei ENDPROC und EXEC auf.	28
PROC ERROR — PROC-FEHLER Es wurde eine unbekannte Prozedur aufgerufen. Dieser Fehler wird oft von Tippfehlern verursacht. Die Prozedurnamen nach EXEC und PROC müssen gleich sein. Kontrollieren Sie das eventuell mit dem DUMP-Befehl.	29
. ERROR — #-FEHLER Es wurde eine unbekannte Marke (Label) benutzt. Sehen Sie sich die Anmerkungen unter Nummer 29 an.	30
BREAK ABORT — MIT BREAK ABGEBROCHEN Es wurde während einer Eingabe/Ausgabe-Operation die BREAK-Taste betätigt und damit die Ausführung abgebrochen.	128
IOCB ALREADY OPEN — KANAL BEREITS GEÖFFNET Es wurde versucht, einen IOCB (Input/Output Control Block; ein Teil des Speichers, der für die Kommunikation mit einem Gerät reserviert ist) zu öffnen, der bereits geöffnet wurde.	129
NONEXISTENT DEVICE — GERÄT NICHT VORHANDEN Sie haben ein Peripheriegerät angesprochen, daß nicht existiert; zum Beispiel einen nicht angeschlossenen Drucker. Meistens wird vergessen, bei einem Dateinamen die Laufwerkskennung D: miteinzugeben.	130
IOCB WRITE ONLY — KANAL NUR ZUM SCHREIBEN GEÖFFNET Es wurde eine Leseanweisung an ein Gerät gesandt, das entweder nicht schreiben kann (zum Beispiel der Drucker) oder nicht zu diesem Zweck geöffnet war. Der betreffende Kanal kann aber auch auf Wunsch zum Lesen und Schreiben gleichzeitig geöffnet werden.	131
ILLEGAL HANDLER COMMAND — UNZULÄSSIGER BEFEHL Es wurde an ein bestimmtes Gerät eine Anweisung gegeben, die dieses nicht ausführen kann.	132
DEVICE OR FILE NOT OPEN — KANAL NICHT GEÖFFNET Ein angesprochenes Gerät ist nicht geöffnet worden. Es ist zum Beispiel versucht worden, die Befehle PLOT und DRAWTO in einem Textmodus zu verwenden.	133
BAD IOCB NUMBER — KANALNUMMER UNZULÄSSIG Es wurde eine falsche IOCB-Nummer angegeben. In BASIC dürfen die Nummern 1 bis 7 benutzt werden.	134
IOCB ALREADY OPEN — KANAL NUR ZUM LESEN GEÖFFNET Es wurde eine Schreibanweisung an ein Gerät gesandt, daß nicht zu diesem Zweck geöffnet war. Diese Fehlermeldung ist das Gegenteil von Nummer 131.	135
END OF FILE — DATEIENDE MIT EOF Der Rechner hat einen EndOfFile-Block (das sind die letzten 128 Byte einer Datei) vorgefunden, obwohl ihm vorher mitgeteilt wurde, daß die Datei größer sei. Diese Fehlermeldung kann auch vorkommen, wenn man versucht, aus einer nicht geöffneten Datei auf der Diskette zu	136

lesen.

TRUNCATED RECORD — DATEI ABGESCHNITTEN — 137

Es wurde der Versuch unternommen, einen Datensatz zu lesen, der grösser ist, als der im Betriebssystem (CIO) festgelegte Wert für einen solchen Datensatz (in BASIC 119 Byte, sonst 256). Tritt auch auf, wenn versucht wird, mit PUT abgelegte Daten mit INPUT zu lesen.

DEVICE TIMEOUT — GERÄT ANTWORTET NICHT — 138

Ein angesprochenes Gerät reagiert nicht in der vom Betriebssystem festgelegten Zeit. Es kann auch sein, daß das verkehrte Gerät angesprochen oder eine falsche Gerätenummer benutzt wurde. Andere Ursachen können sein:
-Gerät defekt
-Gerät nicht in Betrieb
-falsche Laufwerksnummer
Wenn der Fehler bei der Benutzung einer Kassettenstation auftritt, kann es sein, daß die Baudrate nicht richtig oder das Band an einer falschen Stelle war. Wenn ein solcher Fehler öfter auftritt, ist es ratsam, zu prüfen, ob alle Kontakte in Ordnung sind. Wenn der Fehler dann immer noch auftritt, sollte das betreffende Gerät gewartet werden; möglicherweise ist ein Bauteil nicht in Ordnung.

DEVICE NAK (Not Acknowledged) — DATENVERKEHR GESTÖRT — 139

Ein Gerät reagiert nicht, da es unverständliche oder unzulässige Befehle empfängt. Kontrollieren Sie alle Kontakte zum Gerät und die von Ihnen gegebenen Befehle. Die Ursache dieses Fehlers ist in starkem Masse abhängig von dem jeweiligen Gerät. Dessen Gebrauchsanweisung ist oftmals die einzige Möglichkeit, etwas über die genauen Ursachen zu erfahren.

SERIAL BUS ERROR — LESEFEHLER ÜBER DEN SERIELLEN BUS — 140

Irgendetwas kann mit dem POKEY-Chip in ihrem Rechner nicht in Ordnung sein. Dadurch wird die Datenübertragung zwischen dem Gerät und dem Rechner unmöglich. Falls dieser Fehler öfters vorkommt, ist mit hoher Wahrscheinlichkeit Ihr Rechner oder das Gerät reparaturbedürftig. Nur bei Verwendung eines Kassettenlaufwerkes besteht noch die Chance, den Fehler durch richtiges Positionieren des Bandes zu beheben (Bit 7 von SKSTAT im POKEY ist gesetzt).

CURSOR OUT OF RANGE — SCHREIBMARKE AUSSERHALB DES BEREICHES — 141

Es ist der Versuch unternommen worden, die Schreibmarke außerhalb des Bildschirmbereichs eines bestimmten Grafikmodus zu positionieren. Das kommt oft vor, wenn man sich im falschen Grafikmodus befindet. Wählen Sie die Werte für die X- und Y-Koordinaten kleiner und kontrollieren Sie Ihre POSITION-, PLOT- und DRAWTO-Anweisungen.

SERIAL BUS OVERRUN — DATENAUSTAUSCH GESTÖRT — 142

Für diesen Fehler gilt in etwa das gleiche, wie für Fehler 140. Der POKEY-Chip kann die Datenflut nicht schnell genug verarbeiten (Bit 5 von SKSTAT im POKEY ist gesetzt).

SERIAL BUS CHECKSUM ERROR — PRÜFSUMMEN-FEHLER — 143

Dies ist eine unklare Fehlermeldung. Die Ursache kann sowohl in der Hard- als auch in der Software liegen. Die von einem Gerät mitübertragene Prüfsumme stimmt nicht mit dem vom Computer berechneten Wert überein. Dieser Fehler ist der häufigste Kassettenfehler. Meistens hilft schon ein erneuter Versuch oder die Reinigung des Schreib/Lesekopfes. Es ist auch sinnvoll, wenn Sie das Band von Hand straffen.

DEVICE DONE — GERÄT NICHT ANSPRECHBAR Ein Befehl kann von dem angesprochenen Gerät nicht ausgeführt werden. Mögliche Ursachen sind: -es wurde versucht, auf eine schreibgeschützte Diskette zu schreiben. -es wurde versucht, auf einen beschädigten Sektor zu schreiben. -eine zu lesende Diskette ist unformatiert. Und so weiter.	144
ILLEGAL SCREEN MODE — UNZULÄSSIGER GRAFIKMODUS Es wurde eine Grafikbetriebsstufe mit unzulässiger Nummer angesprochen. Die Ursache kann auch sein, daß beim Schreiben auf eine Diskette die Verifizierung einen Fehler meldet. In diesem Fall wurde auf die Diskette nicht das geschrieben, was hätte geschrieben werden sollen.	145
FUNCTION NOT IMPLEMENTED — FUNKTION NICHT VORGESEHEN Es wurde einem Gerät eine Anweisung gegeben, die nicht für dieses bestimmt sein kann. Zum Beispiel: -aus dem Drucker lesen. -auf die Tastatur schreiben.	146
INSUFFICIENT RAM — ZUWENIG SPEICHER FÜR EINEN GRAFIKMODUS Es steht zuwenig freier Speicherplatz für eine angewählte Grafikstufe zur Verfügung (beim ATARI 600 XL zum Beispiel in Modi mit sehr hoher Auflösung).	147
DRIVE NUMBER ERROR — FALSCHE LAUFWERKSNUMMER Die gewählte Laufwerksnummer liegt nicht zwischen 1 und 8 oder das betreffende Diskettenlaufwerk ist nicht angemeldet beziehungsweise eingeschaltet.	160
TOO MANY OPEN FILES — ZU VIELE OFFENE KANÄLE Es ist kein Puffer mehr frei, um einen weiteren Datenkanal zu öffnen. Es dürfte kein Problem sein, die momentan nicht benötigten Dateien (Kanäle) zu schließen. In DOS 2.x kann man mit dem Befehl POKE 1801,anzahl die höchste Anzahl der zu öffnenden Dateien verändern. Mit der Option H des DOS-Menüs können Sie das dann festlegen.	161
DISK FULL — DISKETTE VOLL Es können keine weiteren Dateien mehr auf die Diskette geschrieben werden. Dieser Fehler wird festgestellt, während eine Datei gerade geschrieben wird. Diese Datei ist also nicht vollständig auf der Diskette abgelegt worden. Es muß also eine neue Diskette benutzt werden, auf die die Datei nocheinmal von neuem geschrieben werden muß. Die unvollständige Datei steht auf der vollen Diskette im Inhaltsverzeichnis wie eine normale Datei. Löschen Sie diese Datei, um a) wieder Platz zu gewinnen für eine kleinere Datei und b) nicht irrtümlich mit der unvollständigen Datei zu arbeiten.	162
UNRECOVERABLE SYSTEM I/O ERROR — ENDGÜLTIGER DATENVERLUST Fehler im Diskettenbetriebssystem. Versuchen Sie, ein anderes DOS zu verwenden.	163
FILE NUMBER MISMATCH — FEHLERHAFTE SEKTORVERKETTUNG In jeder Datei gibt es zwei wichtige Nummern. Die erste ist die sogenannte SECTOR LINK (Sektorverbindung), die auf den jeweils nächs-	164

ten Sektor der Datei zeigt. Die zweite ist die FILENUMBER (Dateinummer). Diese gibt es in jedem Sektor. Wenn eine dieser Nummern in einem bestimmten Sektor verkehrt ist, (dies wird auch oft zum Schutz von kommerziellen Programmen gemacht) kann dieser Fehler auftreten. Drastische Maßnahmen, wie die Behandlung durch ein Diskdoctor-Proramm können manchmal helfen. Die Erfolgsquote liegt aber durchaus nicht bei 100 Prozent. Wenn ein solches Disktool ("Werkzeug") nicht hilft, ist die betreffende Datei verloren.

Fehler	Bedeutung	Nr.
FILE NAME ERROR	FALSCHE ZEICHEN IM DATEINAMEN	165

Im Dateinamen und/oder im Anhang wurden unzulässige Zeichen benutzt oder das erste Zeichen eines Dateinamens ist kein Großbuchstabe. Das erste Zeichen darf auf keinen Fall eine Zahl sein, ferner dürfen keine Kleinbuchstaben, Grafikzeichen oder Leerschritte im Namen vorkommen.

Fehler	Bedeutung	Nr.
POINT DATA LENGTH ERROR	POINT-ANGABEN FALSCH	166

Die Länge eines Datensatzes ist nicht korrekt.

Fehler	Bedeutung	Nr.
FILE PROTECTED	DATEI GESICHERT	167

Eine gesicherte Datei sollte gelöscht beziehungsweise überschrieben werden. Entsichern Sie diese Datei erst.

Fehler	Bedeutung	Nr.
DEVICE COMMAND INVALID	UNZULÄSSIGER BEFEHL	168

Das angesprochene Gerät hat einen unzulässigen Befehl erhalten.

Fehler	Bedeutung	Nr.
DIRECTORY FULL	INHALTSVERZEICHNIS VOLL	169

Das Inhaltsverzeichnis enthält bereits das Maximum von 64 Einträgen. Dies hat nichts mit dem zur Verfügung stehenden Speicherplatz auf der Diskette zu tun (siehe Nummer 162).

Fehler	Bedeutung	Nr.
FILE NOT FOUND	DATEI NICHT GEFUNDEN	170

Es wurde eine unbekannte Datei aufgerufen. Prüfen Sie die betreffende Dateispezifikation, da diese exakt der im Inhaltsverzeichnis der Diskette entsprechen muß. DIR oder die Wahl A im DOS-Menü können hier weiterhelfen.

Fehler	Bedeutung	Nr.
POINT INVALID	UNGÜLTIGER POINT-BEFEHL	171

Es ist ein POINT-Befehl eingegeben worden, dessen Anzahl der Bytes nicht der Anzahl der Bytes in der Datei entspricht.

Fehler	Bedeutung	Nr.
ILLEGAL APPEND	DISKETTE MIT FALSCHEM DOS	172

Ein sehr seltener Fehler, der auftritt, wenn man DOS 2 verwendet, um Dateien von DOS 1 zu öffnen, um etwas anzufügen (APPEND). Kopieren Sie die betreffende Datei erst auf eine DOS 2-Diskette und wiederholen Sie den Befehl dann.

Fehler	Bedeutung	Nr.
BAD SECTORS FORMAT TIME	FORMATIERUNGS-FEHLER	173

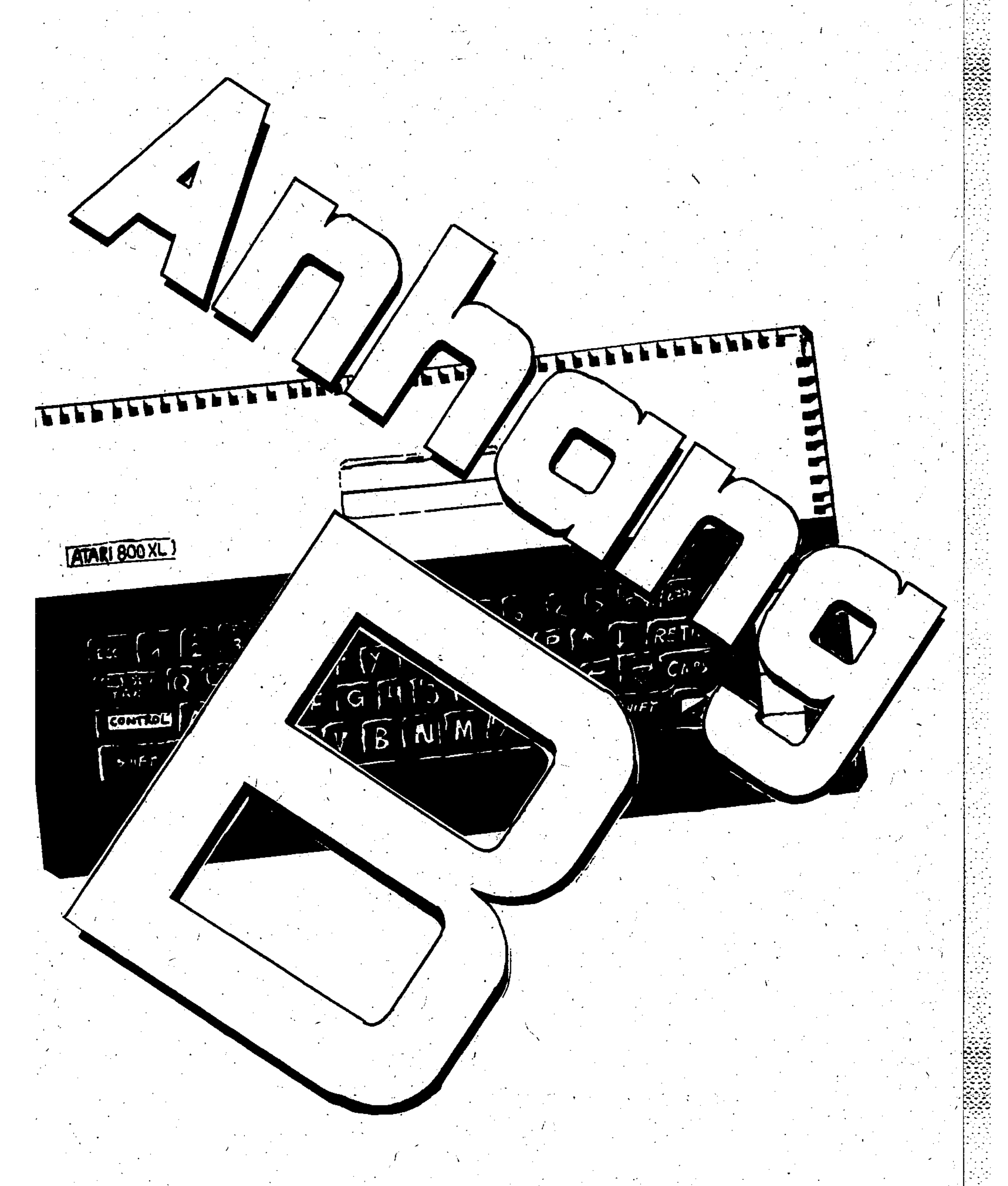
Das Laufwerk hat beim Formatieren mechanische Defekte der Diskette (BAD SECTORS) festgestellt. Sie sollten eine andere Diskette verwenden. Kommt dieser Fehler öfter vor, sollte die Diskettenstation gewartet werden (die Hilfsprogramme Archiver und Disk Wizard II haben die Möglichkeit, solche BAD SECTORS zu formatieren).

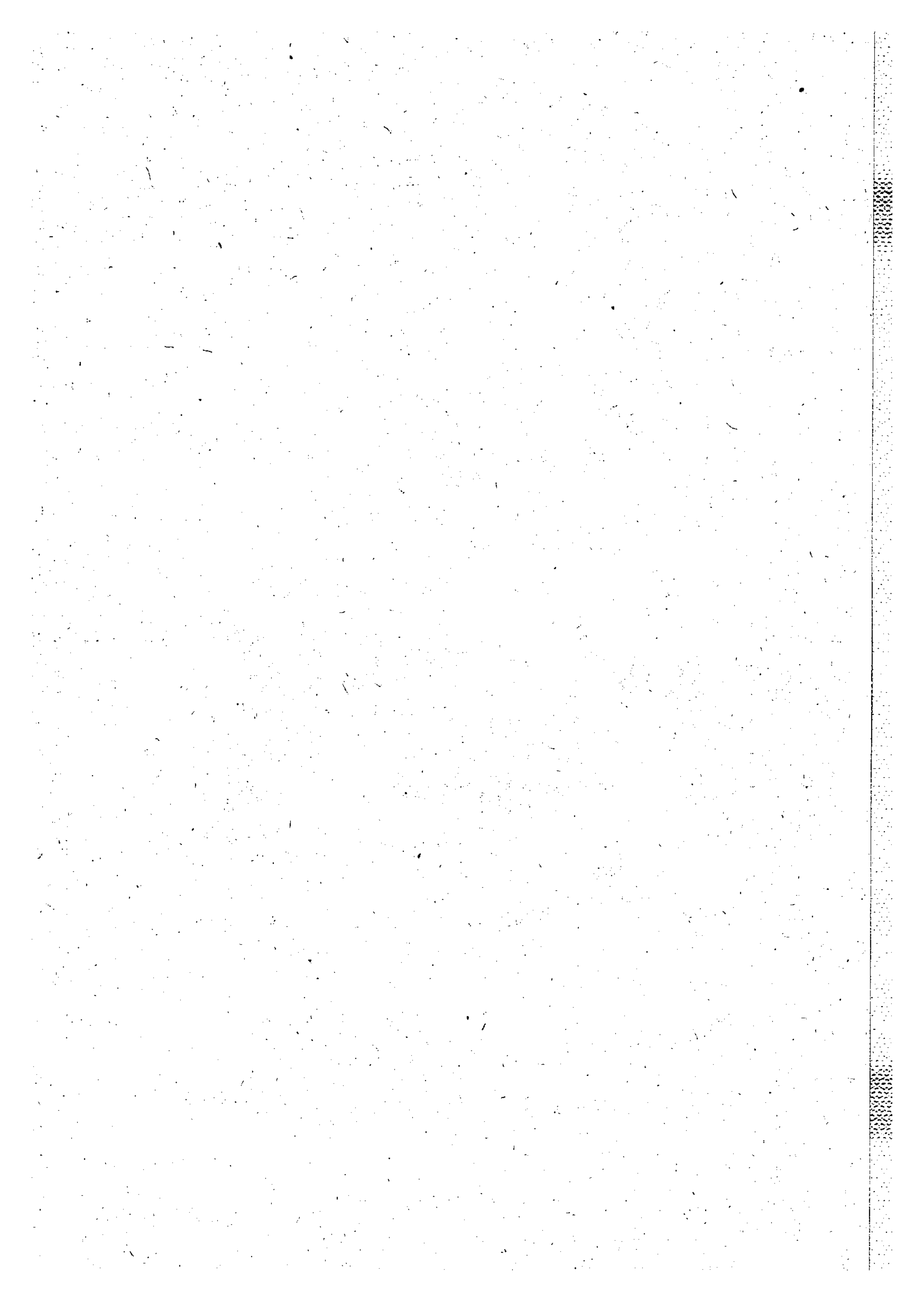
DUPLICATE FILENAME — DOPPELTER DATEINAME Es wurde versucht, mittels einer RENAME-(Umbenennen)Operation zwei Dateien auf einer Diskette denselben Namen zuzuweisen. Das Laufwerk weigert sich, Ihren Befehl auszuführen. Sollten Sie auf die eine oder andere Weise doch einmal zu zwei Dateien mit gleichem Namen gelangen, so können Sie diesen Fehler mit dem Programm DISKFIX.COM wieder beheben. Es befindet sich auf der DOS 2.5 Masterdiskette. Wählen Sie nach dem Laden die Option "Rename by file#".	174
BAD LOAD FILE — DATEI KANN NICHT GELADEN WERDEN Es ist etwas mit einer Datei nicht in Ordnung, wodurch das Laufwerk nicht in der Lage ist, sie zu laden. Der Ausdruck "BAD LOAD FILE" bedeutet normalerweise, daß ein BASIC-Programm geladen werden sollte, wo ein Maschinensprache-Programm erwartet wurde (mit Option L aus dem DOS-Menü oder mit BRUN in Turbo BASIC).	175
INCOMPATIBLE FORMAT — FALSCHES FORMAT Es wurde versucht, ein DOS 3-Diskette von DOS 2 anzusprechen oder umgekehrt. Benutzen Sie ein entsprechendes Wandlungsprogramm.	176
DISK STRUCTURE DAMAGED — DISKETTENSTRUKTUR BESCHÄDIGT Die Diskette wurde durch mechanische oder elektrische Einflüsse beschädigt und kann nicht mehr gelesen werden. Möglich ist aber auch, daß sich auf der Diskette ein "exotisches" Format befindet, daß vom DOS nicht richtig erkannt wird.	177

Die obenstehenden Fehlermeldungen bekommen in Turbo BASIC XL 1.5 noch eine zusätzliche Dimension. Hier wird die Fehlermeldung mit einem erklärenden Text ausgegeben, der das Nachsuchen in solchen Listen oft überflüssig macht. Das sieht dann so aus:

```
ERROR-22? NEST
ERROR-29? PROC
ERROR- 2? MEM
```

und so weiter.
Falls trotz dieser Fehlermeldungen und der vorangegangenen Liste immer noch Fragen offen bleiben, hilft das Handbuch des betreffenden Gerätes sicher weiter. Auf Fehlermeldungen, die nur im Zusammenhang mit einem bestimmten Gerät entstehen, weist das betreffende Handbuch meist besonders hin.
Und lassen Sie sich nicht von dieser ellenlangen Liste entmutigen!


Anhang
B
ATARI 800 XL



ANHANG B: INPUT und OUTPUT mit der Diskette in BASIC

Dieser Anhang beschreibt die "Handgriffe", die nötig sind, um Eingabe- und Ausgabeoperationen mit der Diskette zu programmieren. Er enthält eine Liste vom Befehlen, die in diesem Zusammenhang wichtig sind. Verschiedene Beispiele und Querverweise zu Beispielen an anderen Stellen in diesem Handbuch versuchen, die Verwendung vom Datenbeständen auf Diskette so transparent wie möglich zu machen.

Um eine gewisse Einheit bei der Bezeichnung von Dateien zu erreichen, folgt hier eine Auflistung der meistgebrauchten Extensionen. Durch sie soll deutlich gemacht werden, um welche Art Datei es sich handelt.

```
*.BAS - BASIC-Programme (ATARI BASIC)
*.CTB - Compilierte BASIC-Programme
*.COM - Maschinensprache-Programme
*.EXE -            "
*.OBJ -            "
*.SYS - Systemprogramme
*.LST - mit LIST gespeicherte BASIC-Programme
*.DAT - Datenbestände
*.FNT - Zeichensätze
*.TXT - Texte
*.TUR - BASIC-Programme (Turbo BASIC)
*.PIC - Bilder im Maltafel-Format
*.MIC - Bilder im Micropainter-Format
```

Sie sind bei der Wahl ihrer Dateinamen natürlich frei; aber um einem Dschungel von bizarren Namen zu vermeiden, ist es ratsam, den Dateinamen so gut wie möglich auf die Datei abzustimmen. Dies dient nicht nur Ihrer eigenen Übersicht.

Befehle

Unten sehen Sie eine Liste mit I/O-Befehlen, die im Hauptteil 3 näher beschrieben werden.

SAVE dateispezifikation
Schreibt ein Programm aus dem Speicher auf die Diskette.

LIST dateispezifikation
Siehe SAVE, aber im ATASCII-Format.

LOAD dateispezifikation
Liest ein Programm von der Diskette in den Speicher.

ENTER dateispezifikation
Siehe LOAD, aber im ATASCII-Format. Dieser Befehl dient dem Zusammenfügen von Programmen.

RUN dateispezifikation
Liest ein Programm von der Diskette in den Speicher und führt es aus.

DELETE dateispezifikation
Löscht eine Datei von der Diskette.

RENAME dateispezifikation,dateispezifikation
Benennt eine Datei um.

LOCK dateispezifikaion
Schützt eine Datei auf der Diskette vor Löschen, aber nicht vor dem Formatieren.

UNLOCK dateispezifikation
Hebt den Schutz von LOCK auf.

BLOAD dateispezifikation
Siehe LOAD, aber für eine binäre Datei (Maschinensprache).

BRUN dateispezifikation
Siehe RUN, aber für eine binäre Datei (Maschinensprache).

DIR dateispezifikation
Ruft das Inhaltsverzeichnis der Diskette auf.

Befehle, die gebraucht werden, um Datenbestände auf der Diskette zu schaffen:

BGET	BPUT
%GET	%PUT
GET	PUT
OPEN #	CLOSE #
NOTE	POINT
INPUT #	PRINT #

Um eine sequentielle Datei auf der Diskette zu schaffen und Daten in ihr abzulegen, müssen folgende Schritte ausgeführt werden:

1. öffnen der Datei für Ausgabe (OUTPUT) oder Anfügen (APPEND), mittels des OPEN-Befehls.

2. Schreiben der Daten in die Datei mittels
PRINT #, BPUT, PUT oder %PUT.

3. Schließen der Datei mit CLOSE #.

Das folgende Beispiel veranschaulicht die oben beschriebenen Schritte:

```
10 OPEN #1,8,0,"D:DATEI.DAT"    Schritt 1
20 PRINT #1,A$                  Schritt 2
30 PRINT #1;B$                  Schritt 2
40 CLOSE #1                     Schritt 3
```

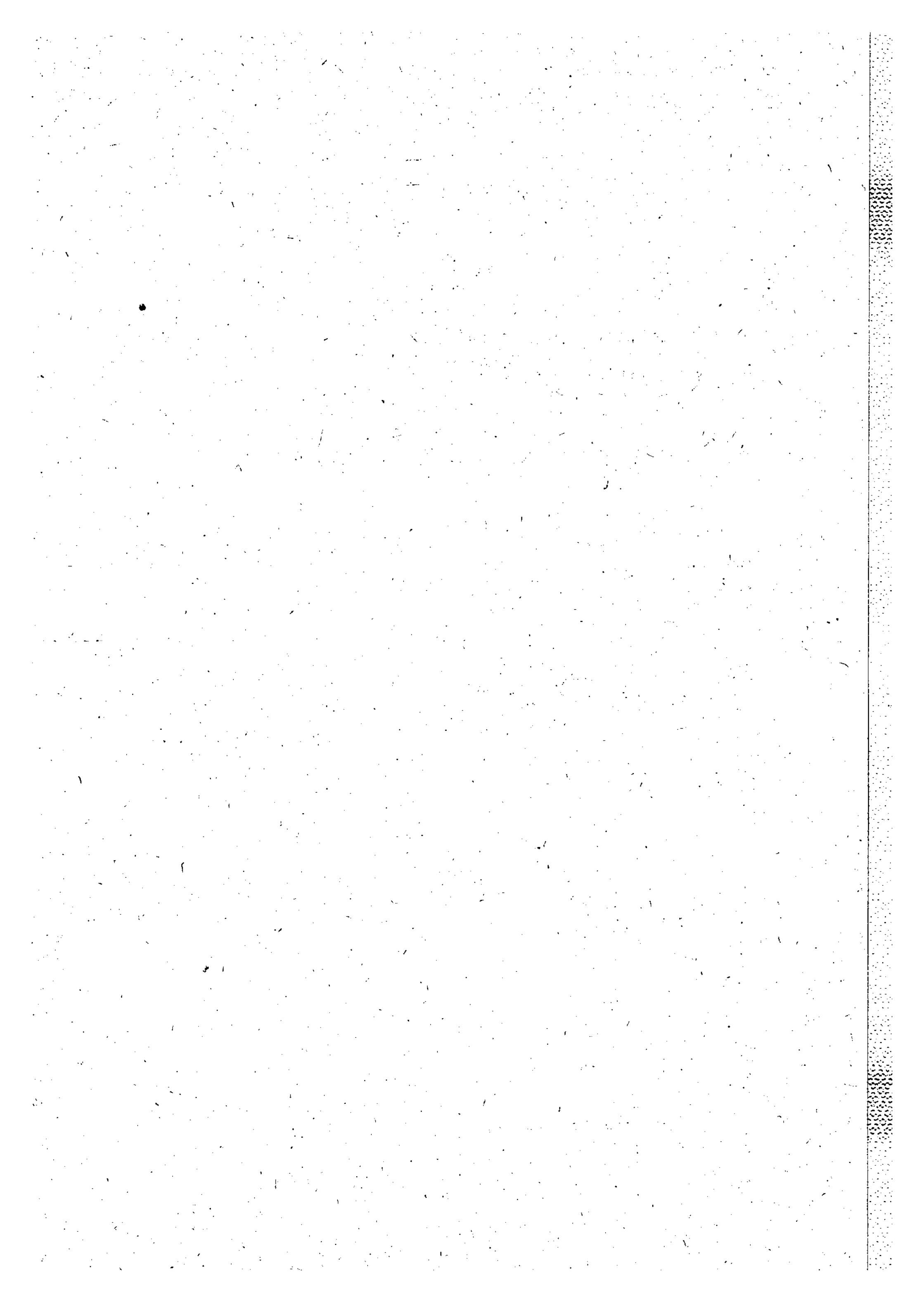
Das Lesen der Daten erfolgt mittels:
INPUT #, BGET, GET oder %GET

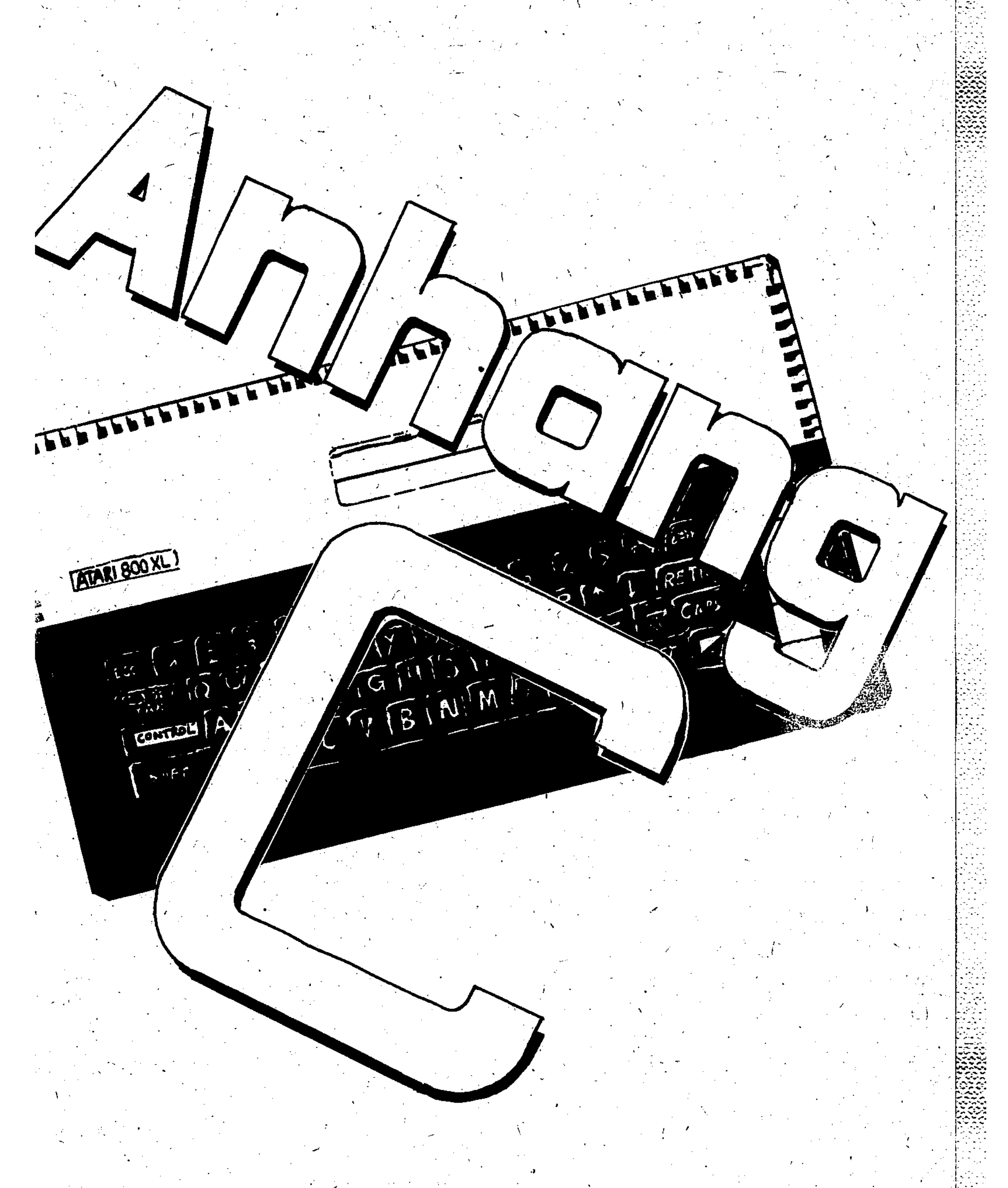
Beispiel:

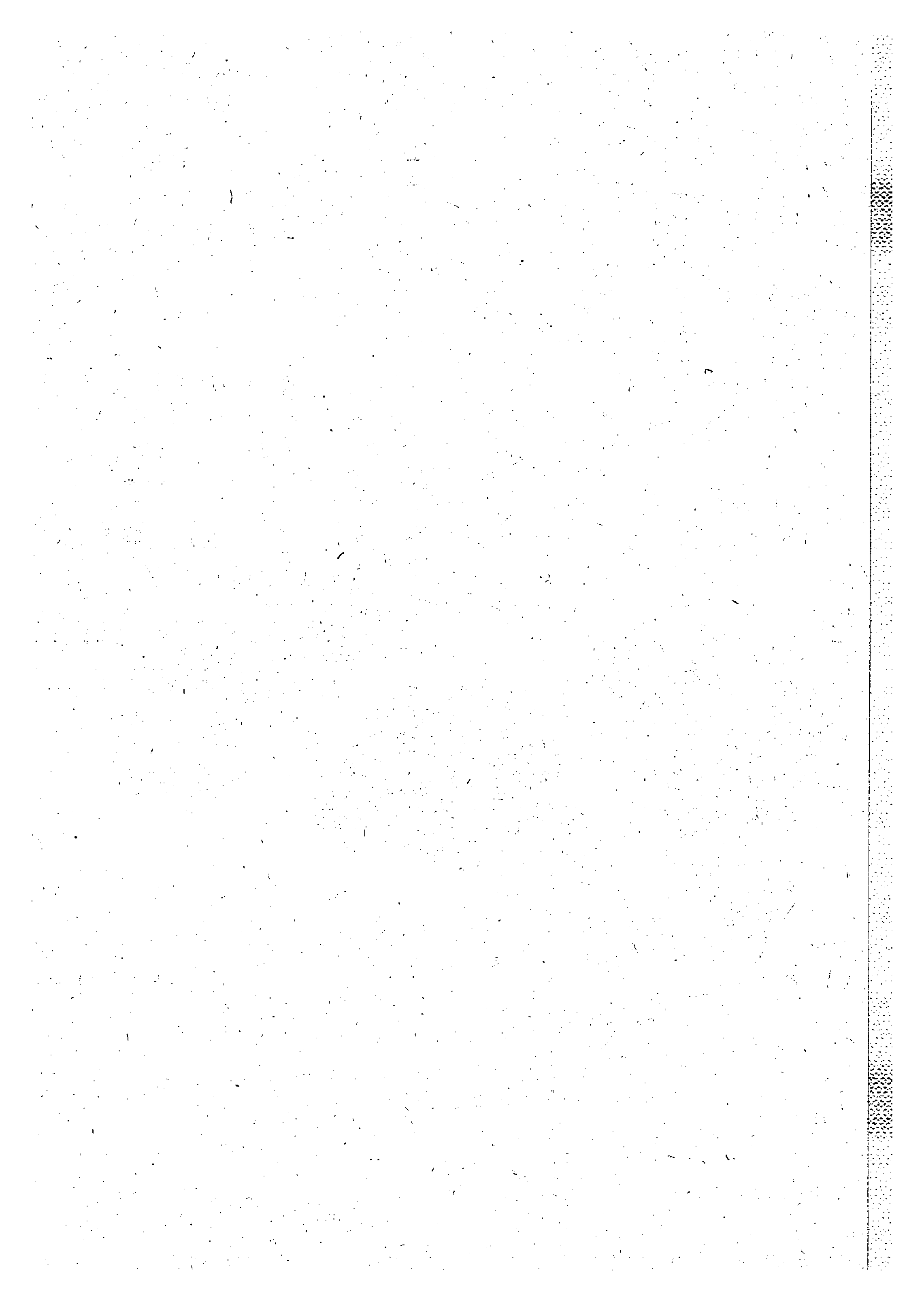
```
    10 OPEN #1,4,0,"D:DATEI.DAT     Schritt 1
    20 INPUT #1,X$,Y$               Schritt 2
    30 CLOSE #1                     Schritt 3
```

Wenn Sie mehr Informationen über die betreffenden Befehle haben möchten, sehen Sie sich bitte Hauptteil 3 an.
Das folgende Beispiel legt eine Datei von über die Tastatur eingegebenen Daten an.

```
    10 DIM NAME$(20),S$(20),WO$(20)
    20 INPUT "Name ";NAME$: OPEN #1,8,0,"D:DATEI.DAT"
    30 WHILE NAME$<>"ENDE"
    40   INPUT "Strasse & Nr. ";S$
    50   INPUT "Wohnort       ";WO$
    60   GOSUB 1000
    70   INPUT "Name          ";NAME$
    80 WEND
    90 CLOSE
    100 TRAP 170
    110 OPEN #1,4,0,"D:DATEI.DAT"
    120 INPUT #1,NAME$,S$,WO$
    130 PRINT NAME$
    140 PRINT S$
    150 PRINT WO$
    160 GOTO 120
    170 CLOSE
    180 END
    1000 PRINT #1;NAME$
    1010 PRINT #1;S$
    1020 PRINT #1;WO$
    1030 RETURN
```




Anhang
C
ATARI 800 XL
CONTROL



ANHANG C: ATASCII-Zeichen und Tastaturcodes

Dieser Anhang enthält eine Liste mit den Zeichen, die im Festwertspeicher der ATARI Computer abgelegt sind. Diese Zeichen werden auch ATASCII-Codes genannt.

Die Tabelle gibt den dezimalen Wert, das entsprechende ASCII-Zeichen (falls vorhanden) und zeigt, welche Taste(n) gedrückt werden müssen, um dieses Zeichen zu erzeugen.

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
0		NULL	CTRL-,	11		VT	CTRL-K
1		SOH	CTRL-A	12		FF	CTRL-L
2		STX	CTRL-B	13		CR	CTRL-M
3		ETX	CTRL-C	14		SO	CTRL-N
4		EOT	CTRL-D	15		SI	CTRL-O
5		ENQ	CTRL-E	16		DLE	CTRL-P
6		ACK	CTRL-F	17		DC1	CTRL-Q
7		BEL	CTRL-G	18		DC2	CTRL-R
8		BS	CTRL-H	19		DC3	CTRL-S
9		HT	CTRL-I	20		DC4	CTRL-T
10		LF	CTRL-J	21		NAK	CTRL-U

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
22		SYN	CTRL-V	42	*	*	*
23		ETB	CTRL-W	43	+	+	+
24		CAN	CTRL-X	44	,	,	,
25		EM	CTRL-Y	45	-	-	-
26		SUB	CTRL-Z	46	.	.	.
27		ESC	ESC\ESC	47	/	/	/
28		FS	ESC\CTRL--	48	0	0	0
29		GS	ESC\CTRL-=	49	1	1	1
30		RS	ESC\CTRL-+	50	2	2	2
31		US	ESC\CTRL-*	51	3	3	3
32		Leerzeichen	Leertaste	52	4	4	4
33	!	!	SHIFT-1	53	5	5	5
34	"	"	SHIFT-2	54	6	6	6
35	#	#	SHIFT-3	55	7	7	7
36	$	$	SHIFT-4	56	8	8	8
37	%	%	SHIFT-5	57	9	9	9
38	&	&	SHIFT-6	58	:	:	SHIFT-;
39	'	'	SHIFT-7	59	;	;	;
40	(	(	SHIFT-9	60	<	<	<
41	)	)	SHIFT-0	61	=	=	=

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
62	>	>	>	82	R	R	R
63	?	?	SHIFT-/	83	S	S	S
64	@	@	SHIFT-8	84	T	T	T
65	A	A	A	85	U	U	U
66	B	B	B	86	V	V	V
67	C	C	C	87	W	W	W
68	D	D	D	88	X	X	X
69	E	E	E	89	Y	[illegible]	[illegible]
70	F	F	F	90	Z	[illegible]	[illegible]
71	G	G	G	91	[	[	SHIFT- ;
72	H	H	H	92	\	\	SHIFT- ,
73	I	I	I	93	]	]	SHIFT- +
74	J	J	J	94	^	↑	SHIFT- *
75	K	K	K	95	_	←	SHIFT–
76	L	L	L	96	♦	[illegible]	CTRL-.
77	M	M	M	97	a	[illegible]	(LOWR) A
78	N	N	N	98	b	b	(LOWR) B
79	O	O	O	99	c	c	(LOWR) C
80	P	P	P	100	d	d	(LOWR) D
81	Q	Q	Q	101	e	e	(LOWR) E

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
102		f	(LOWR) F	122		z	(LOWR) Z
103		g	(LOWR) G	123		{	CTRL-;
104		h	(LOWR) H	124		|	SHIFT-=
105		i	(LOWR) I	125	[6]	}	ESC\CTRL-< oder ESC\SHIFT-<
106		j	(LOWR) J	126	[7]	~	ESC\BACK S
107		k	(LOWR) K	127	[8]	DEL	ESC\TAB
108		l	(LOWR) L	128			(Λ) CTRL-,
109		m	(LOWR) M	129			(Λ) CTRL-A
110		n	(LOWR) N	130			(Λ) CTRL-B
111		o	(LOWR) O	131			(Λ) CTRL-C
112		p	(LOWR) P	132			(Λ) CTRL-D
113		q	(LOWR) Q	133			(Λ) CTRL-E
114		r	(LOWR) R	134			(Λ) CTRL-F
115		s	(LOWR) S	135			(Λ) CTRL-G
116		t	(LOWR) T	136			(Λ) CTRL-H
117		u	(LOWR) U	137			(Λ) CTRL-I
118		v	(LOWR) V	138			(Λ) CTRL-J
119		w	(LOWR) W	139			(Λ) CTRL-K
120		x	(LOWR) X	140			(Λ) CTRL-L
121		y	(LOWR) Y	141			(Λ) CTRL-M

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
142			(Λ) CTRL-N	162	"		(Λ) SHIFT-2
143			(Λ) CTRL-O	163	#		(Λ) SHIFT-3
144			(Λ) CTRL-P	164	$		(Λ) SHIFT-4
145			(Λ) CTRL-Q	165	%		(Λ) SHIFT-5
146			(Λ) CTRL-R	166	&		(Λ) SHIFT-6
147			(Λ) CTRL-S	167	'		(Λ) SHIFT-7
148			(Λ) CTRL-T	168	(		(Λ) SHIFT-9
149			(Λ) CTRL-U	169	)		(Λ) SHIFT-0
150			(Λ) CTRL-V	170	*		(Λ) *
151			(Λ) CTRL-W	171	+		(Λ) +
152			(Λ) CTRL-X	172	,		(Λ) ,
153			(Λ) CTRL-Y	173	-		(Λ) -
154			(Λ) CTRL-Z	174	.		(Λ) .
155	EOL		(Λ) RETURN	175	/		(Λ) /
156			ESC\SHIFT-BACK S	176	0		(Λ) 0
157			ESC\SHIFT->	177	1		(Λ) 1
158			ESC\CTRL-TAB	178	2		(Λ) 2
159			ESC\SHIFT-TAB	179	3		(Λ) 3
160			(Λ) Leertaste	180	4		(Λ) 4
161	!		(Λ) SHIFT-1	181	5		(Λ) 5

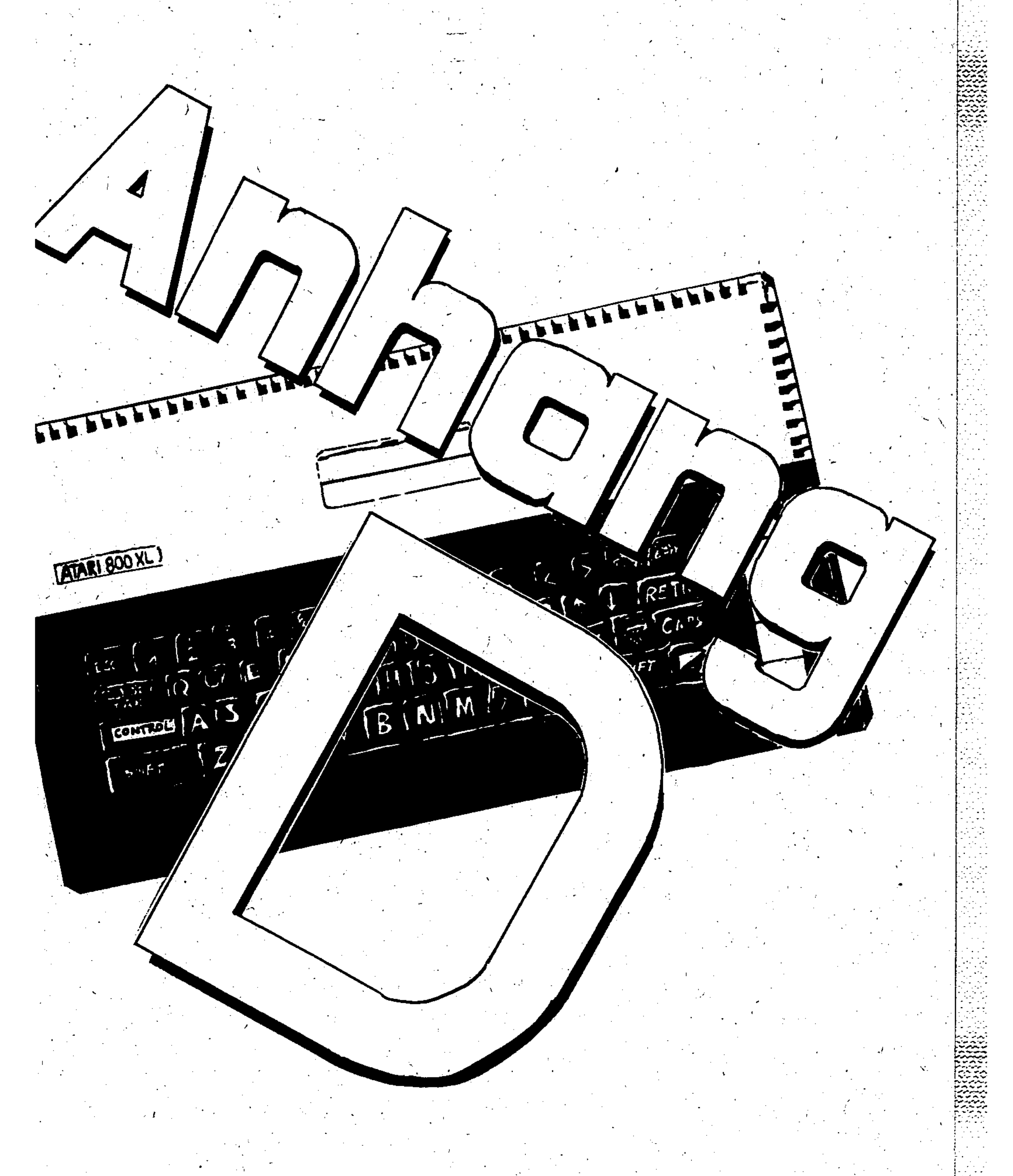
Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
182	6		(人) 6	202	J		(人) J
183	7		(人) 7	203	K		(人) K
184	8		(人) 8	204	L		(人) L
185	9		(人) 9	205	M		(人) M
186	:		(人) SHIFT-;	206	N		(人) N
187	;		(人) ;	207	O		(人) O
188	<		(人) <	208	P		(人) P
189	=		(人) =	209	Q		(人) Q
190	>		(人) >	210	R		(人) R
191	?		(人) SHIFT-/	211	S		(人) S
192	@		(人) SHIFT-8	212	T		(人) T
193	A		(人) A	213	U		(人) U
194	B		(人) B	214	V		(人) V
195	C		(人) C	215	W		(人) W
196	D		(人) D	216	X		(人) X
197	E		(人) E	217	Y		(人) Y
198	F		(人) F	218	Z		(人) Z
199	G		(人) G	219	[		(人) SHIFT-,
200	H		(人) H	220	\		(人) SHIFT-+
201	I		(人) I	221	]		(人) SHIFT-.

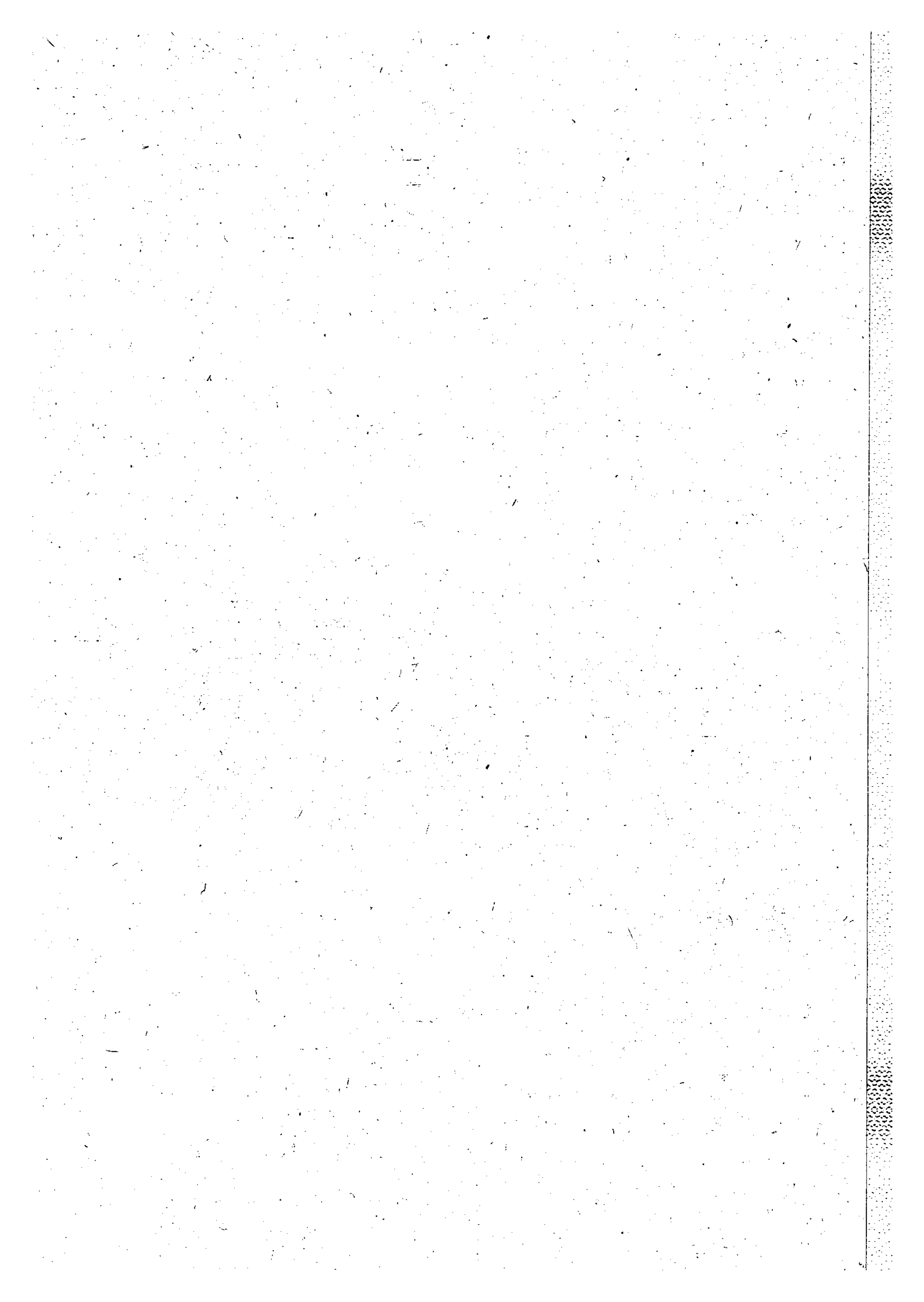
Dezimalkode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimalkode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
222	^		(Λ) SHIFT-*	239	o		(Λ) (LOWR) O
223	_		(Λ) SHIFT–	240	p		(Λ) (LOWR) P
224	◆		(Λ) CTRL-.	241	q		(Λ) (LOWR) Q
225	a		(Λ) (LOWR) A	242	r		(Λ) (LOWR) R
226	b		(Λ) (LOWR) B	243	s		(Λ) (LOWR) S
227	c		(Λ) (LOWR) C	244	t		(Λ) (LOWR) T
228	d		(Λ) (LOWR) D	245	u		(Λ) (LOWR) U
229	e		(Λ) (LOWR) E	246	v		(Λ) (LOWR) V
230	f		(Λ) (LOWR) F	247	w		(Λ) (LOWR) W
231	g		(Λ) (LOWR) G	248	x		(Λ) (LOWR) X
232	h		(Λ) (LOWR) H	249	y		(Λ) (LOWR) Y
233	i		(Λ) (LOWR) I	250	z		(Λ) (LOWR) Z
234	j		(Λ) (LOWR) J	251	♠		(Λ) CTRL-;
235	k		(Λ) (LOWR) K	252	|		(Λ) SHIFT-=
236	l		(Λ) (LOWR) L	253	↰ [14]		ESC\CTRL-2
237	m		(Λ) (LOWR) M	254	◀ [15]		(Λ) ESC\ CTRL-BACK S
238	n		(Λ) (LOWR) N	255	▶ [16]		(Λ) ESC\CTRL->

Unten sehen Sie eine Tabelle, in der die Codes stehen, die generiert werden, wenn eine Taste auf der Tastatur gedrückt wird. Diese Werte sind nicht dieselben, wie die Vorangegangenen.

Taste	Code	+Shift Code	+Ctrl Code
A	63	127	191
B	21	85	149
C	18	82	146
D	58	122	186
E	42	106	170
F	56	120	184
G	61	125	189
H	57	121	185
I	13	77	141
J	1	65	129
K	5	69	133
L	0	64	128
M	37	101	165
N	35	99	163
O	8	72	136
P	10	74	138
Q	47	111	175
R	40	104	168
S	62	126	190
T	45	109	173
U	11	75	139
V	16	80	144
W	46	110	174
X	22	86	150
Y	43	107	171
Z	23	87	151

Taste Taste	Code	+Shift Code	+Ctrl Code
0	50	114	178
1	31	95	-
2	30	94	158
3	26	90	154
4	24	88	152
5	29	93	157
6	27	91	155
7	51	115	179
8	53	117	181
9	48	112	176
;	2	66	130
+	6	70	134
*	7	71	135
=	15	79	143
<	54	118	182
>	55	119	183
ESC	28	92	156
SPC	33	97	161
TAB	44	108	172
INV	39	103	167
CAPS	60	124	188
DEL	52	116	180
RET	12	76	140


Anhang
D
ATARI 800 XL



Die Bedeutung und Herkunft der einzelnen Ausdrücke in BASIC:

Ausdruck	Abgeleitet vom englischen:	Deutsche Entsprechung:
ABS	ABSolute	absolut
ADR	ADResse	Adresse
ASC	Atari Standard Code	Atari-Standardverschlüsselung
ATN	ArcTaNgent	Arcus Tangens
*B	Break	Unterbrechung
BGET	Block GET	aufnehmen in Blöcken
BLOAD	Binary LOAD	binäres Laden
BPUT	Block PUT	ablegen in Blöcken
BRUN	Binary RUN	binäres Starten
BYE	BYE	Verabschiedung
CHR$	Character string	Zeichen(kette)
CIRCLE	CIRCLE	Kreis
CLOAD	Cassette LOAD	von der Kassette laden
CLOG	Common LOGarithm	Logarithmus zur Basis 10
CLOSE	CLOSE	schließen
CLR	CLeaR	bereinigen
CLS	CLear Screen	den Bildschirm bereinigen
COLOR	COLOR	Farbe
CONT	CONTinue	fortsetzen
COS	COSine	Cosinus
CSAVE	Cassette SAVE	auf die Kassette speichern
DATA	DATA	Daten
DEC	DECimal value	dezimaler Wert
DEG	DEGree	Grad
DEL	DELete (line)	tilgen (Zeile)
DELETE	DELETE (diskette)	tilgen (Diskette)
DIM	DIMension	Dimension, Ausmaß
DIR	DIRectory	Inhaltsverzeichnis
DIV	DIVision	Disvision, Teilung
DO	DO	tun
DOS	Diskette Operating System	Disketten-Betriebssystem
DPEEK	Double PEEK (Memory)	doppelt nachsehen (im Speicher)
DPOKE	Double POKE (Memory)	doppelt hineinlegen (in den Speicher)
DRAWTO	DRAW TO	bis zu ... zeichnen
DSOUND	Double SOUND	doppelt erklingen, doppelter Ton
DUMP	DUMP (variables)	(Variablen) ausschütten
ELSE	ELSE	sonst
END	END	beenden
ENIF	END of IF	IF beenden
ENDPROC	END of PROCedure	Prozedur beenden
ENTER	ENTER (program)	(ein Programm) einbringen
ERR	ERRor (code)	Fehler-Verschlüsselung
ERL	ERror Line	Fehlerzeile
EXEC	EXECute	ausführen
EXIT	EXIT	Ausgang
EXP	EXPonent	Exponent
*F	test For-next	FOR-NEXT testen
FCOLOR	Fill COLOR	Füllfarbe
FILLTO	FILL TO	zu ... füllen
FOR	FOR	für
FRAC	FRACtion	Bruchstück
FRE	FREe	frei

GET	GET	aufnehmen
%GET	GET	aufnehmen
GO#	GO label	zur Marke gehen
GOSUB	GO to SUBroutine	zum Unterprogramm gehen
GOTO	GO TO	zu ... gehen
HEX$	HEXadecimal string	hexadezimale Zeichenkette
IF	IF	wenn, falls
INKEY$	string IN KEYstroke	Zeichenkette im Tastendruck
INPUT	INPUT	eingeben
INSTR	string IN STRing	Zeichenkette in Zeichenkette
INT	INTeger	integer, ganzzahlig
*L	List tabulator	Tabulator für die Programmliste
LEN	LENgth	Länge
LET	LET	lassen
LIST	LIST	auflisten
LOAD	LOAD	laden
LOCATE	LOCATE	lokalisieren, ausfindig machen
LOCK	LOCK	verschließen
LOG	LOGarithm	natürlicher Logarithmus
LOOP	LOOP	Schleife
LPRINT	Lister PRINT	über den Drucker schreiben, ausgeben
MOD	MODulus	Modulo
MOVE	MOVE	bewegen
NEW	NEW	neu
NEXT	NEXT	nächstes
NOTE	NOTE	notieren, merken
ON	ON	auf
OPEN	OPEN	öffnen
PADDLE	PADDLE	Drehregler
PAINT	PAINT	malen
PAUSE	PAUSE	pausieren
PEEK	PEEK	nachsehen
PLOT	PLOT	eintragen, einzeichnen
POINT	POINT	auf ... richten
POKE	POKE	hineinlegen
POP	POP (a subroutine)	(ein Unterprogramm) verpuffen lassen
POSITION	POSITION	Position, Lage
PRINT	PRINT	drucken, schreiben, ausgeben
PROC	PROCedure	Prozedur, Unterprogramm
PTRIG	Paddle TRIGger	Feürknopf am Drehregler
PUT	PUT	ablegen
%PUT	PUT	ablegen
RAD	RADians	Bogen
RAND	RANDom	Zufall
READ	READ	lesen
REM	REMark	bemerken, hinzufügen
RENAME	RENAME	umbennenen
RENUM	RENUMber	umnumerieren
REPEAT	REPEAT	wiederholen
RESTORE	RESTORE	wiedereinsetzen, wiederherstellen
RETURN	RETURN (from a subroutine)	(von einem Unterprogramm) zurückkehren
RND	RaNDom	Zufall
RUN	RUN	laufen, rennen, starten
SAVE	SAVE	sichern, speichern
SETCOLOR	SET a COLOR	eine Farbe setzen
SGN	SiGN	(Vor-)Zeichen
SIN	SINe	Sinus
SOUND	SOUND	erklingen, Klang

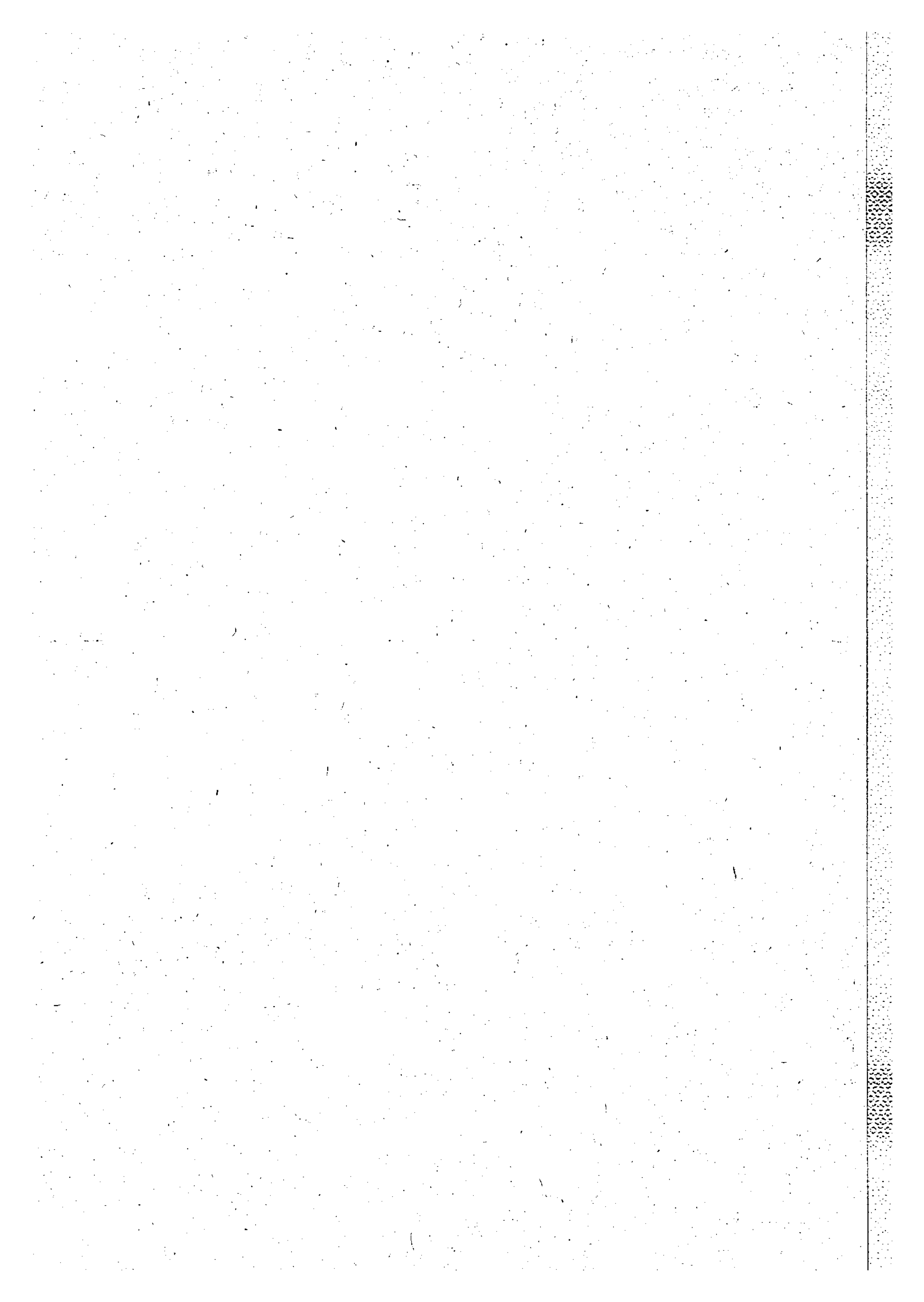
SQR	SQare Root	Quadratwurzel
STATUS	STATUS	Status, Zustand
STEP	STEP	Schritt
STICK	joySTICK	Steuerhebel
STOP	STOP	stoppen, halten
STR$	STRing string	Zahl als Zeichen(kette)
STRIG	joyStick TRIGrigger	Feuerknopf am Steuerhebel
TEXT	TEXT	Text
THEN	THEN	dann
TIME	TIME	Zeit
TIME$	TIME string	Zeit in einer Zeichenkette
TRACE	TRACE	nachspüren, verfolgen
TRAP	TRAP	ertappen, in eine Falle gehen
TRUNC	TRUNCate	abschneiden
UINSTR	Uppercase string IN STRing	Zeichenkette in Zeichenkette ohne Rücksicht auf Großbuchstaben
UNLOCK	UNLOCK	aufschließen
UNTIL	UNTIL	bis
USR	User SubRoutine	Anwender-Unterprogramm in Maschinensprache
VAL	VALUE	Wert
WEND	END While	WHILE beenden
WHILE	WHILE	während, solange

Häufig benutzte Ausdrücke im Zusammenhang mit Computern:

Adresse .... Nummer einer Speicherstelle
Algorithmus .... Rechenvorgang mit sich wiederholendem Schema
Append (engl.)...... anhängen
Array (engl.)....... Variablenfeld
ACSII (engl.)....... American Standard Code for Information Interchange
Amerikanische Standardverschlüsselung für Datenaustausch
ATASCII (engl.)..... ATARI Standard Code for Information Interchange
ATARI Standardverschlüsselung für Datenaustausch
BASIC (engl.)....... Beginner's All Purpose Symbolic Instruction Code
Allzweck-Programmiersprache für Anfänger
Bit (engl.)......... kleinste Informationseinheit
Break (engl.)....... Programmunterbrechung
Byte (engl.)........ Informationseinheit (=8 Bit)
Cassette (engl.).... Kassette
Charakter (engl.)... Zeichen
Code (engl.)........ Verschlüsselung
Command (engl.)..... Befehl
Compiler (engl.).... Programm, das alle auszuführenden Instruktionen mit einem mal in Maschinensprache übersetzt
Computer (engl.).... Elektronenrechner
Cursor (engl.)...... Schreibmarke
Debug (engl.)....... Fehler beseitigen
Density (engl.)..... Dichte
Destination (engl.). Bestimmung, Ziel
Device (engl.)...... Gerät
Dezimalsystem ...... Zahlensystem mit der Basis 10
Directory (engl.)... Inhaltsverzeichnis
Display (engl.)..... Anzeige
Disk Drive (engl.).. Diskettenlaufwerk
DOS-Menue (engl.)... Programm, das die Auswahl von DOS-Funktionen zuläßt

Drive (engl.)....... Laufwerk
Editor (engl.)...... Bildschirmbearbeitungsgerät, Schreibprogramm
EOF (engl.)......... End Of File (Ende der Datei) (Tastenkombination CONTOL-3)
EOL (engl.)......... End Of Line (Ende der Zeile) (RETURN-Taste)
Error (engl.)....... Fehler
Expression (engl.).. Ausdruck
Extensionen (engl.).... Anhang
Extension (engl.)... Ausdehnung
False (engl.)....... unwahr
File (engl.)........ Datei
Frequenz . ... Schwingungen pro Zeiteinheit
Handler (engl.)..... Programm, das dem Rechner den Umgang mit einem Gerät erklärt
Hardware (engl.).... Computer und Peripheriegeräte
Hexadezimalsystem .. Zahlensystem mit der Basis 16
I/O (engl.)......... Eingabe/Ausgabe
Initialisierung .... Vorbereitung
Invers .... umgekehrt, also Weiß auf Schwarz statt umgekehrt
Illegal .... nicht erlaubt
Input (engl.)....... Eingabe
Interface (engl.)... Verbindungsstück
Interpreter (engl.). Programm, das auszuführende Befehle jeweils in Maschinensprache übersetzt und ausführt
IOCB (engl.)........ Input/Output Control Block, Eingabe/Ausgabe-Kontrollblock
Key (engl.)......... Taste
Keyboard (engl.).... Tastatur
Label (engl.)....... Marke, die zum Wiederfinden einer Stelle dient, Name
Legal .... erlaubt
Manual (engl.)...... Handbuch
Memory (engl.)...... Speicher
Message (engl.)..... Mitteilung, Meldung
Modus .... Art und Weise
MS .... Maschinensprache
Nested (engl.)...... verschachtelt
Option .... Wahlmöglichkeit
Output (engl.)...... Ausgabe
Overflow (engl.).... mehr Informationen als zulässig
Port (engl.)........ "Hafen", Schnittstelle zur Außenwelt, Buchse
Printer (engl.)..... Drucker
RAM (engl.)......... Random Access Memory (wahlfreier Speicher)
Read (engl.)........ Lesen
ROM (engl.)......... Read Only Memory (Festwertspeicher)
Runtime (engl.)..... Laufzeit eines des Programms
Screen (engl.)...... Bildschirm
Software (engl.).... Programme und Daten (Information)
Source (engl.)...... Quelle
Space (engl.)....... Raum, meist ist die Leerschrittaste gemeint
Spezifikation ...... nähere Benennung, Beschreibung
Stack (engl.)....... Stapelspeicher
Statement (engl.)... Anweisung
String (engl.)...... Variable vom Typ String (enthält Zeichen(ketten))
Subroutine (engl.).. Unterprogramm
Syntax (engl.)...... Satzbau, vorgeschriebene Folge der Zeichen
Tape (engl.)........ Kassettenband
Timeout (engl.)..... nach einer festgelegten Zeit keine Meldung erhalten
True (engl.)........ wahr
TV (engl.).......... Television, Fernsehgerät

Update (engl.)...... der neuste Stand, auf diesen bringen
User (engl.)........ Anwender, Anwenderin
Verify (engl.)...... Überprüfung
Volume (engl.)...... Lautstärke
Wildcard (engl.).... Platzhalter, Joker (bei Dateinamen mit ähnlichen Namen)
Window (engl.)...... Fenster
Write (engl.)....... Schreiben





Stichwort-
verzeichnis
ATARI 1010
RECORD
PLAY
ADVANCE

